TUGAS AKHIR – TE 141599

ALGORITMA MENGHADANG BOLA DENGAN METODE *FUZZY LOGIC* UNTUK ROBOT PENJAGA GAWANG SEPAK BOLA BERODA

Kamal Arief
NRP 07111440000194

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – TE141599

ALGORITMA MENGHADANG BOLA DENGAN METODE *FUZZY LOGIC* UNTUK ROBOT PENJAGA GAWANG SEPAK BOLA BERODA

Kamal Arief
NRP 07111440000194

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TE141599

ALGORITHM FACING BALL WITH FUZZY LOGIC METHOD FOR WHEELED GOAL KEEPER ROBOT

Kamal Arief
NRP 07111440000194

Supervisor
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

PERNYATAAN KEASLIAN

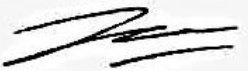
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Algoritma Menghadang Bola Dengan Metode *Fuzzy Logic* untuk Robot Penjaga Gawang Sepak Bola Beroda" adalah benar-benar hasil karyaintelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis dalam daftar pustaka.

Apabila ternyata pernyataan ini tidak benar maka saya siap menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 12 Juli 2017



Kamal Arief

NRP 0711144000194

ALGORITMA MENGHADANG BOLA DENGAN METODE *FUZZY LOGIC* UNTUK ROBOT PENJAGA GAWANG SEPAK BOLA BERODA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Djoko Purwanto, M.Eng.	**Dr. Ir. Hendra Kusuma, M.Eng.Sc**
NIP: 196512111990021002	**NIP: 196409021989031003**

SURABAYA
Juli, 2018

ALGORITMA MENGHADANG BOLA DENGAN METODE *FUZZY LOGIC* UNTUK ROBOT PENJAGA GAWANG SEPAK BOLA BERODA

Nama : Kamal Arief
Pembimbing I : Dr. Ir. Djoko Purwanto, M.Eng.
Pembimbing II : Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ABSTRAK

Robot sepak bola beroda adalah suatu robot yang bekerja secara tim untuk melakukan permainan sepak bola. Peran penjaga gawang dalam permainan ini sangat vital karena merupakan pertahanan terakhir sebelum mendapatkan gol. Oleh karena itu, perlu penjagaan yang responsif dalam menghadang bola. Untuk mendapatkan penghadangan bola yang responsif metode *Fuzzy Logic* dapat diterapkan disini.

Input *Fuzzy Logic* dalam penelitian ini berupa sudut bola terhadap robot dan kecepatan pergerakan bola. Dengan mengetahui sudut bola terhadap robot kita akan mendapatkan orientasi pergerakan robot tersebut menuju bola. Lalu, dengan mengetahui kecepatan bola kita akan mendapatkan kecepatan robot yang diperlukan untuk menghadang bola dengan tepat. Robot penjaga gawang pada tugas akhir kali ini menggunakan robot dari Tim Robot Sepak Bola Beroda ITS. Pengujian dalam penelitian ini untuk mengetahui respon robot dalam menghadang bola pada setiap tendangan.

Pada hasil pengujian yang dilakukan robot telah berhasil menghadang bola dengan tepat, dari 50 percobaan tendangan, robot ini dapat menghadang 38 tendangan. Persentase penyelamatan yaitu 76%. Dari data sudut bola, kecepatan pergerakan bola dan keluaran dari metode *fuzzy* sesuai dengan rule yang direncanakan.

Kata kunci : robot penjaga gawang, menghadang bola, *Fuzzy Logic*

Halaman ini sengaja dikosongkan

ALGORITHM FACING BALL WITH FUZZY LOGIC METHOD FOR WHEELED GOAL KEEPER ROBOT

Name : Kamal Arief
Supervisor : Dr. Ir. Djoko Purwanto, M.Eng.
Co-Supervisor : Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ABSTRACT

A wheeled soccer robot is a robot that works in a team to do soccer games. The role of the goalkeeper in this game is vital as it is the last defense before getting a goal. Therefore, there needs to be a responsive guard in blocking the ball. To obtain a responsive ball retouch the Fuzzy Logic method can be applied here.

Fuzzy Logic input in this research is the angle of the ball against the robot and the speed of movement of the ball. By knowing the angle of the ball against the robot we will get the orientation of the movement of the robot to the ball. Then, by knowing the speed of the ball we will get the robot speed required to block the ball properly. The goalkeeper robot in this final project uses a robot from ITS Wheeled Soccer Robot Team. Testing in this research is to know the robot response in blocking the ball and the percentage of rescue that can be done in several kick.

In the test results conducted robots have managed to block the ball properly, from 50 kicks experiment, this robot can block 38 kicks. The percentage of rescue is 76%. From the data angle of the ball, the speed of movement of the ball and the output of the fuzzy method in accordance with the planned rule.

Keywords: robot goalkeeper, blocking ball, Fuzzy Logic, response

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul: Algoritma Menghadang Bola Dengan Metode *Fuzzy Logic* untuk Robot Penjaga Gawang Sepak Bola Beroda.

Tugas Akhir ini merupakan persyaratan dalam menyelesaikan pendidikan program Strata-Satu di jurusan Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dibuat berdasarkan teori–teori yang didapatkan selama mengikuti perkuliahan, berbagai literatur penunjang dan pengarahan dosen pembimbing dari awal hingga akhir pengerjaan tugas akhir ini.

Pada kesempatan ini, penulis ingin berterimakasih kepada pihak-pihak yang membantu pembuatan tugas akhir ini, khususnya kepada:

1. Bapak, Ibu, adik serta seluruh keluarga yang memberikan dukungan baik moril maupun materiil.
2. Dr. Ir. Djoko Purwanto, M.Eng. selaku dosen pembimbing 1 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
3. Dr. Ir. Hendra Kusuma, M.Eng.Sc. selaku dosen pembimbing 2 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
4. Ir. Tasripan, M.T. selaku Koordinator Bidang Studi Elektronika.
5. Dr.Eng. Ardyono Priyadi, S.T., M.Eng. selaku Ketua Jurusan Teknik Elektro ITS Surabaya.
6. Seluruh dosen bidang studi elektronika.
7. Teman-teman laboratorium Elektronika yang tidak dapat disebutkan satu-persatu, telah membantu proses pengerjaan tugas akhir ini.

Kesempurnaan hanya milik Allah SWT, penyusunan tugas akhir ini tentu masih banyak kekurangan. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat terutama untuk tim IRIS ITS. Semoga dengan penelitian ini performa dari robot sepakbola ITS meningkat.

Surabaya, 27 April 2018

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

ABSTRAK ..... **i**
ABSTARCT ..... **iii**
KATA PENGANTAR ..... **v**
DAFTAR ISI ..... **vii**
CONTENS ..... **xi**
DAFTAR GAMBAR ..... **xv**
DAFTAR TABEL ..... **xix**
BAB 1 PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Metodologi 2
1.6 Sistematika Penulisan 3
1.7 Relevansi dan Manfaat 3
BAB 2 TINJAUAN PUSTAKA DAN TEORI PENUNJANG **5**
2.1 Logika Fuzzy 5
2.1.1 Himpunan Fuzzy 5
2.1.2 Fungsi Keangotaan 5
2.1.3 Operator Fuzzy 8
2.1.4 Penalaran Monoton 8
2.1.5 Fungsi Implikasi 8
2.1.6 Sistem Inferensi Fuzzy 9
2.1.7 Metode Tsukamoto 9
2.2 OpenCV 10
2.3 Kamera Omnidireksional 11
2.4 Regresi Polinomial 12
2.5 Open Framework 12
2.6 Baterai Lipo 12
2.7 STM32f4 Discovery 13
2.8 Rotary Encoder 14
2.9 Mini PC 15
2.10 Driver Motor 15
2.11 Motor DC 16
BAB 3 PERANCANGAN SISTEM **17**
3.1 Vision 17
3.2 Algoritma Jalan 22
3.3 Kontrol Pergerakan 22

3.3.1 Perancangan Sistem Gerak 23
3.3.2 Perancangan Sistem Odometri 26
3.4 Desain mekanik robot 27
3.5 Desain elektronik robot 27
3.5.1 Supply 28
3.5.2 Mini PC Intel Nuc 29
3.5.3 STM32F4-Discovery 29
3.5.4 USB to Serial Converter 32
3.5.5 Arduino Nano 32
3.5.6 Sensor IMU 33
3.5.7 Sensor garis 33
3.5.8 Buck Converter untuk Mini PC, Mikrokontroler ... 34
3.5.9 Buck Converter untuk Mini PC 35
3.5.10 Driver Motor 35
3.5.11 Rotary encoder 36
3.5.12 Sensor IR Range Finder 37
3.6 Perancangan Fuzzy Logic 38
3.6.1 Perancangan Fungsi Keanggotaan 39
3.6.2 Perancangan Rule 40
BAB 4 PENGUJIAN 45
4.1 Uji Jarak Bola 45
4.2 Pengujian Jalan Algoritma 46
4.2.1 Data Tendangan Pertama 46
4.2.1 Data Tendangan Kedua 48
4.2.1 Data Tendangan Ketiga 49
BAB 5 KESIMPULAN DAN SARAN 53
5.1 Kesimpulan 53
5.2 Saran 54
DAFTAR PUSTAKA 55
LAMPIRAN 57
BIODATA PENULIS 81

CONTENS

ABSTRAK ..... **i**
ABSTARCT ..... **iii**
PREFACE ..... **v**
DAFTAR ISI ..... **vii**
CONTENS ..... **xi**
LIST OF FIGURES ..... **xv**
LIST OF TABLES ..... **xix**
***CHAPTER* 1 PREFACE** **1**
1.1 *Background* 1
1.2 *Problem Formulation* 2
1.3 *Limitation Problem* 2
1.4 *Objectives* 2
1.5 *Methodology* 2
1.6 *Writing Systematics* 3
1.7 *Relevance and Benefits* 3
CHAPTER 2 BASIC THEORY **5**
2.1 *Fuzzy Logic* 5
2.1.1 *The Fuzzy Set* 5
2.1.2 *Membership Function* 5
2.1.3 *Fuzzy Operator* 8
2.1.4 *Monotonic Reason* 8
2.1.5 *Implication Function* 8
2.1.6 *Fuzzy Inference System* 9
2.1.7 *Tsukamoto Method* 9
2.2 OpenCV 10
2.3 *Omnidirectional Camera* 11
2.4 *Polynomial Regression* 12
2.5 Open Framework 12
2.6 *Lipo Battery* 12
2.7 STM32f4 Discovery 13
2.8 *Rotary Encoder* 14
2.9 Mini PC 15
2.10 Motor Driver 15
2.11 Motor DC 16
CHAPTER 3 SYSTEM PLANNING **17**
3.1 *Vision* 17
3.2 *Moving Algorithm* 22
3.3 *Control of Movement* 22

3.3.1 *Design of Motion System* 23
3.3.2 *Design of Odomety System* 26
3.4 *Mechanical Design of Robot* 27
3.5 *Electronic Design Robot* 27
3.5.1 *Supply* 28
3.5.2 Mini PC Intel Nuc 29
3.5.3 STM32f4 29
3.5.4 USB to Serial Converter 32
3.5.5 Arduino Nano 32
3.5.6 *IMU Sensor* 33
3.5.7 *Line Sensor* 33
3.5.8 *System Regulator of Microcontroller and Sensor* 34
3.5.9 *Buck Converter for Mini PC* 35
3.5.10 *Motor Driver* 35
3.5.11 *Rotary encoder* 36
3.5.12 Sensor IR proximity 37
3.6 *Fuzzy Logic Design* 38
3.6.1 *Design of Membership Function* 39
3.6.2 *Rule Design* 40
CHAPTER 4 EXPERIMENT 45
4.1 Distance Ball Test 45
4.2 Moving Algorithm Test 46
4.2.1 First Kick Test 47
4.2.1 Second Kick Test 48
4.2.1 Third Kick Test 49
CHAPTER 5 CONCLUSION AND SUGGESTION 53
5.1 Conclusion 53
5.2 Suggestion 54
REFERENCES 55
APPENDIX 57
AUTHOR BIODATA 81

DAFTAR GAMBAR

Gambar 2.1.1 Representasi Linear Naik 5
Gambar 2.1.2 Representasi Linear Turun 5
Gambar 2.1.3 Representasi Segitiga 6
Gambar 2.1.4 Inferensi Fuzzy dengan Metode Tsukamoto 9
Gambar 2.2 Logo OpenCV[4]..... 10
Gambar 2.3 Kamera Omnidireksional 10
Gambar 2.5 Logo openFrameworks..... 11
Gambar 2.6 Konfigurasi Baterai Lipo..... 12
Gambar 2.7 STM32F4 - Discovery..... 13
Gambar 2.8 Rotary Encoder..... 13
Gambar 2.9 Mini PC 14
Gambar 2.10 Rangkaian H-Bridge..... 15
Gambar 2.11 Motor DC 15
Gambar 3.1 Blok Diagram Sistem 17
Gambar 3.1.1 Blok Diagram Vision..... 18
Gambar 3.1.2 Hasil Pendeteksian Bola..... 18
Gambar 3.1.3 Hasil Threshold Lapangan(kiri) dan Bola(kanan) 19
Gambar 3.1.4 Ilustrasi Mendapat Kecepatan Bola..... 21
Gambar 3.2.1 Blok Diagram Algoritma Perencaan Gerak..... 22
Gambar 3.3.1 Blok Kontrol Robot..... 23
Gambar 3.3.2 Motor DC PG-45..... 24
Gambar 3.3.3 Roda Omni 24
Gambar 3.3.4 Interkoneksi antara Motor, Driver Motor dan STM32f4 25
Gambar 3.3.5 *Base* robot..... 25
Gambar 3.3.6 Desain base untuk rotary encoder..... 26
Gambar 3.4.1 Desain 3D Robot..... 27
Gambar 3.5.1 Blok diagram sistem elektronik robot 28
Gambar 3.5.1.1 Baterai Lipo..... 28
Gambar 3.5.2.1 Mini PC Intel NUC 5i7 KYK..... 29
Gambar 3.5.3.1 Diagram kontrol PID pada motor..... 30
Gambar 3.5.3.2 Tampilan IDE Atollic..... 31
Gambar 3.5.3.3 Tampilan Software ST-LINK..... 31
Gambar 3.5.4.1 USB to Serial Converter..... 32
Gambar 3.5.5.1 Arduino nano..... 32
Gambar 3.5.6.1 MPU-6050 dan pin outnya 33
Gambar 3.5.7.1 Modul Sensor Garis..... 34
Gambar 3.5.8.1 Modul Buck Converter 12v dan 5v 34
Gambar 3.5.8.2 Modul Buck Converter 19v..... 35

Gambar 3.5.9.1 Modul Driver Motor BTN 7970 36
Gambar 3.5.10.1 *Rotary Encoder* 37
Gambar 3.5.10.2 *Wire out* dari rotary encoder 37
Gambar 3.5.11.1 Sensor IR Range Finder 38
Gambar 3.5.11.2 *Wire Out* Sensor IR Range Finder 38
Gambar 3.6.1.1 Fungsi Keangotaan Sudut Bola 39
Gambar 3.6.1.2 Fungsi Keangotaan Kecepatan Bola 40
Gambar 4.1 Uji Pendeteksian Jarak Bola 45
Gambar 4.2.1.1 Plot Sudut Bola terhadap Robot 46
Gambar 4.2.1.2 Plot Kecepatan Bola 46
Gambar 4.2.1.3 Plot *Output Fuzzy* 47
Gambar 4.2.2.1 Plot Sudut Bola terhadap Robot 47
Gambar 4.2.2.2 Plot Kecepatan Bola 48
Gambar 4.2.2.3 Plot *Output Fuzzy* 48
Gambar 4.2.3.1 Plot Sudut Bola terhadap Robot 49
Gambar 4.2.3.2 Plot Kecepatan Bola 49
Gambar 4.2.3.3 Plot *Output Fuzzy* 50

DAFTAR TABEL

Tabel 3.6.2.1 Rule Fuzzy 42
Tabel 3.6.2.2 *Output Fuzzy* 42
Tabel 4.1 Tabel Jarak Bola Terhadap Robot 44

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Robot adalah alat yang dapat membantu banyak kebutuhan manusia secara fisik baik dengan kontrol oleh manusia ataupun secara otomatis dengan program yang telah ditanamkan terlebih dahulu pada robot. Untuk memacu pengembangan pengethuan tentang robot, ada banyak sekali kontes yang diadakan setiap tahunnya dengan tema-tema yang beraneka ragam. Salah satu kontes yang tersebut adalah Kontes Robot Sepak Bola Indonesia kategori beroda.

Salah satu tantangan dari divisi robot ini adalah menghalau bola agar tidak masuk ke gawang. Tugas ini dilakukan oleh robot penjaga gawang. Banyak metode yang dapat disinggung tentang robot penjaga gawang ini. Salah satunya adalah algoritma menghalau bola dan metode untuk menghalaunya. Disini saya akan menggunakan *Fuzzy Logic* setelah menimbang beberapa metode yang pas dalam menghalau bola tersebut.

Dengan menggunakan kamera dengan sudut pandang 360 derajat tentunya akan membuat sudut pandang robot terhadap lingkungan semakin luas sehingga tidak diperlukan lagi terlalu banyak gerakan mekanik untuk mencari bola sehingga robot dapat lebih cepat untuk menentukan keputusan selanjutnya.

Kamera akan menghasilkan citra lingkuangan yang jika diproses lebih lanjut akan mendapatkan posisi bola pada citra. Posisi bola pada citra dapat di representasikan dengan x, y dari bola dengan satuan pixel. Namun posisi x, y bola pada citra tidak linear terhadap posisi sesungguhnya terhadap robot sehingga diperlukan proses pengolahan lebih lanjut untuk mendapatkan posisi bola terhadap robot. Beberapa metode yang dapat digunakan adalah Jaringan Saraf tiruan atau regresi polinomial.

Data posisi bola tersebut akan saya sampling selama waktu tertentu untuk mendapatkan data kecepatan dan sudut bola terhadap robot. Data-data ini yang selanjutnya akan menjadi masukkan *fuzzy* yang kemudian akan masuk ke pengkategorian membership function dalam prosedur *Fuzzification*. Data yang sudah dikelompokkan akan masuk ke *Rule Set* dan akan mengeluarkan suatu *statement* sesuai *rulebase* yang telah

ditentukan yang selanjutnya masuk ke proses *Defuzzification* yang menghasilkan output yang akan masuk proses selanjutnya. Output dari *Fuzzy Logic* tersebut yaiti kecepatan pergerakan robot dan posisi robot terhadap sumbu X lapangan.

1.2 Perumusan Masalah

Permasalahan pada penelitian ini adalah sebagai berikut:

1. Algoritma menghadang bola agar tidak masuk ke gawang.
2. Bagaimana menciptakan sistem penghadangan bola agar tidak masuk ke gawang.

1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Objek yang di deteksi hanyalah berupa bola dengan warna jingga.
2. Pergerakan bola yang diambil hanya berupa kecepatan dan sudut pepindahan bola.
3. Jarak pembacaan posisi bola hanya 4,5 meter diatas lapangan hijau dan tidak tertutupi obstacle berwarna lain.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Membuat rancangan Algoritma penghadangan bola pada robot sepak bola beroda.
2. Menciptakan Sistem penghadangan bola pada robot sepak bola beroda.

1.5 Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap studi literatur dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Dasar teori tersebut diambil dari artikel-artikel di internet dan forum-forum diskusi.

2. Perancangan Sistem

Pada tahap ini akan dibahas bagaimana perancangan dari sistem robot yang akan dibuat dalam tugas akhir ini.

3. **Pengujian dan Penyempurnaan Sistem**
   Pada tahap ini, dilakukan pengujian untuk mengetahui kinerja sistem yang dibuat pada tugas akhir ini

4. **Penulisan Laporan Tugas Akhir**
   Tahap penulisan laporan tugas akhir dilakukan beriringan pengerjaan.

1.6 Sistematika Penulisan

Sistematika pembahasan dari Proyek Akhir ini direncanakan sebagai berikut:

- **BAB 1: PENDAHULUAN**
  Pada Bab ini berisi hal-hal meliputi latar belakang, perumusan masalah, batasan masalah, tujuan, metodelogi, sistematika penulisan serta relevansi dan manfaat.
- **BAB 2: TINJAUAN PUSTAKA DAN TEORI PENUNJANG**
  Bab ini berisi tentang teori penunjang sera literatur dari berbagai sumber baik dari jurnal, buku, atau internet yang dapat menunjang pengerjaan tugas akhir ini.
- **BAB 3: PERANCANGAN SISTEM**
  Bab ini menjelaskan tentang perencanaan sistem baik disisi *hardware* ataupun *software* untuk pengukuran posisi bola terhadap robot.
- **BAB 4: PENGUJIAN**
  Bab ini berisi tentang pengujian sistem yang telah dibuat beserta analisa dari hasil pengujian tersebut.
- **BAB 5: PENUTUP**
  Bab ini adalah bagian yang berisikan kesimpulan yang diperoleh dari pengerjaan tugas akhir serta saran-saran untuk pengembangan riset lebih lanjut.

1.7 Relevansi dan Manfaat

Manfaat yang didapat dari tugas akhir ini adalah dengan meningkatkan kemampuan penjaga gawang dalam menghadang bola agar tidak terjadi gol dan dapat dipakai dalam lomba KRSBI-Beroda.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA DAN TEORI PENUNJANG

2.1 Logika Fuzzy

Logika Fuzzy adalah suatu metode kecerdasan buatan yang dikenalkan pertama kali oleh Lothfi A. Zadeh pada tahun 1965[1]. Inti dari Logika Fuzzy adalah memetakan ruang input ke ruang output. Mekanisme utama untuk melakukannya adalah pernyataan *if-then* yang disebut aturan. Semua aturan dievaluasi secara paralel dan tidak memperhatikan urutannya. Aturan tersebut merujuk pada variabel dan kata sifat yang menggambarkan variabel-variabel tersebut. Sebelum membangun sistem dan menafsirkan aturannya, perlu ditentukan semua istilah yang akan direncanakan untuk digunakan dan kata sifat yang menggambarkan keadaannya. Sebagai contoh, untuk menentukan bahwa robot dapat berjalan "cepat", harus ditentukan kisaran kecepatan robot yang bervariasi dan memenuhi maksud dengan kata "cepat".

2.1.1 Himpunan Fuzzy

Himpunan fuzzy dapat diartikan sebagai suatu kelas bilangan dalam suatu batasan yang samar. Pada himpunan tegas (*crisp*), nilai keanggotaan *x* dalam suatu himpunan **A**, yang sering ditulis dengan $\mu_A[x]$ dan memiliki dua kemungkinan, yaitu : Satu (1), yang berarti bahwa suatu item menjadi angota dalam suatu himpunan atau Nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan. Pada himpunan fuzzy nilai keanggotaan bernilai 0 sampai 1. Himpunan fuzzy memiliki 2 atribut, yaitu :

a. Linguistik

Penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti : Muda, Parobaya, Tua.

b. Numeris

Suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti : 15, 50, 70.

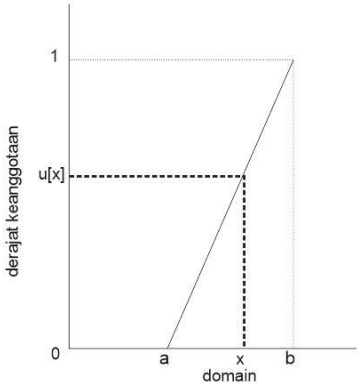
2.1.2 Fungsi Keanggotaan

Fungsi keanggotaan adalah suatu himpunan input data yang direpresentasikan dalam sebuah fungsi yang bernilai 0 sampai 1 atau kata lain fungsi keangotaan merepresentasikan derajart keanggotaan

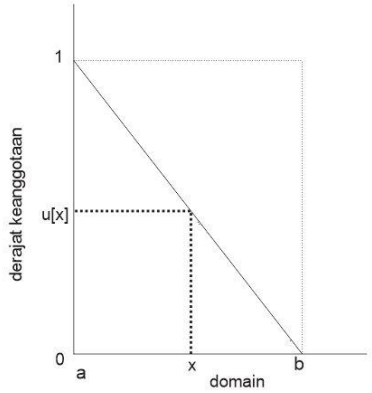
dalam logika fuzzy. Ada beberapa representasi dari fungsi keanggotaan, yaitu:

a. *Representasi linear*

Pada representasi linear, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep fuzzy.



Gambar 2.1.1 Representasi linear naik



Gambar 2.1.2 Representasi linear turun

Pada representasi linear naik, perhitungan derajat keanggotaannya yaitu sebagai berikut:

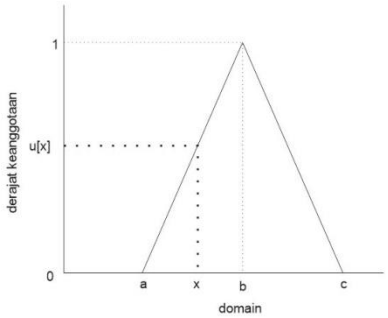
$$\mu[x] = \begin{cases} 0, & lainnya \\ \dfrac{(x-a)}{(b-a)}, & a \le x \le b \\ 1, & x > b \end{cases}$$

Pada representasi linear turun, perhitungan derajat keanggotaannya yaitu sebagai berikut:

$$\mu[x] = f(x) = \begin{cases} 0, & lainnya \\ \dfrac{(b-x)}{(b-a)}, & a \le x \le b \end{cases}$$

b. Representasi segitiga

Representasi segitiga adalah gabungan antara 2 garis linear.



Gambar 2.1.3 Representasi segitiga

Pada representasi segitiga, perhitungan derajat keanggotaannya yaitu sebagai berikut:

$$\mu[x] = \begin{cases} 0, & x < a \; atau \; x > c \\ \dfrac{(x-a)}{(b-a)}, & a \le x \le b \\ \dfrac{(b-x)}{(c-b)}, & b \le x \le c \end{cases}$$

Kita dapat memilih berbagai macam fungsi keanggotaan, namun harus tetap dipilih sesuai kebutuhan kita. Kita harus bisa memperkirakan bahwa sistem butuh nilai yang beragam atau tidak tergantung kebutuhannya.

2.1.3 Operator Fuzzy

Untuk mengkombinasi dan memodifikasi himpunan fuzzy ada beberapa operasi khusus untuk melakukannya. Derajat keanggotaan yang berasal dari hasil operasi 2 himpunan sering dikenal dengan nama *fire-strength* atau ∝-predikat[2]. Ada 3 jenis operator fuzzy, yaitu:

1. *Operator AND*
   Operator *AND* adalah interseksi pada dua himpunan. ∝-predikat diperoleh dengan mengambil nilai derajat keanggotaan terkecil antar himpunan yang bersangkutan
   $$\mu_{A\cap B} = \min(\mu_A[x], \mu_B[y])$$
2. *Operator OR*
   Operator *AND* adalah union pada dua himpunan. ∝-predikat diperoleh dengan mengambil nilai derajat keanggotaan terbesar antar himpunan yang bersangkutan
   $$\mu_{A\cup B} = \max(\mu_A[x], \mu_B[y])$$
3. *Operator NOT*
   Operator *NOT* adalah komplemen pada satu himpunan. ∝-predikat diperoleh dengan mengurangkan nilai derajat keanggotaan suatu himpunan yang bersangkutan dengan 1.
   $$\mu'_A = 1 - \mu_A[x]$$

2.1.4 Penalaran Monoton

Metode penalaran secara monoton digunakan untuk dasar teknik implikasi fuzzy. Jika dua daerah fuzzy direlasikan dengan implikasi sederhana *if A then B.* Maka system fuzzy akan berjalan tanpa melalui fuzzifikasi dan defuzzifikasi. Nilai output dapat diestimasi secara langsung dari nilai keanggotaan berdasar antesedennya[2].

Sebagai contoh, jika tinggi badan "tinggi" maka berat badan akan "berat". Dari tinggi badan tersebut kita cari nilai derajat keanggotaannya, lalu nilai tersebut kita gunakan untuk mencari berat badan dari nilai derajat keanggotaan yang sama.

2.1.5 Fungsi Implikasi

Bentuk umum dari fungsi implikasi adalah *if-then*,sebagai contoh:

IF x is A THEN y is B

Dengan x dan y adalah nilai scalar, dan A dan B adalah himpunan fuzzy. Proporsi yang mengikuti *IF* disebut anteseden, sedangkan yang mengikuti *THEN* disebut konsekuen[2]. Ada dua fungsi implikasi yang dapat digunakan, yaitu:

a. Min(minimum). Fungsi ini akan memotong output himpunan fuzzy
b. Dot(product). Fungsi ini akan menskala output himpunan fuzzy.

2.1.6 Sistem Inferensi Fuzzy

Sistem inferensi fuzzy merupakan proses pengolahan data dalam bentuk crisp input yang melalui beberapa tahapan dalam sistem fuzzy untuk menghasilkan data dalam bentuk crips output. Terdapat tiga metode sistem inferensi fuzzy, yaitu: Mamdani, Sugeno dan Tsukamoto. Tahap sistem inferensi fuzzy yang harus dilalui, yaitu :

a. Nilai Input
Berupa masukan dalam bentuk nilai pasti (crisp).

b. Fuzzifikasi
Proses merubah crisp input menjadi fuzzy menggunakan fungsi keanggotaan, setiap variabel fuzzy dimodelkan ke dalam fungsi keanggotaan yang dipilih.

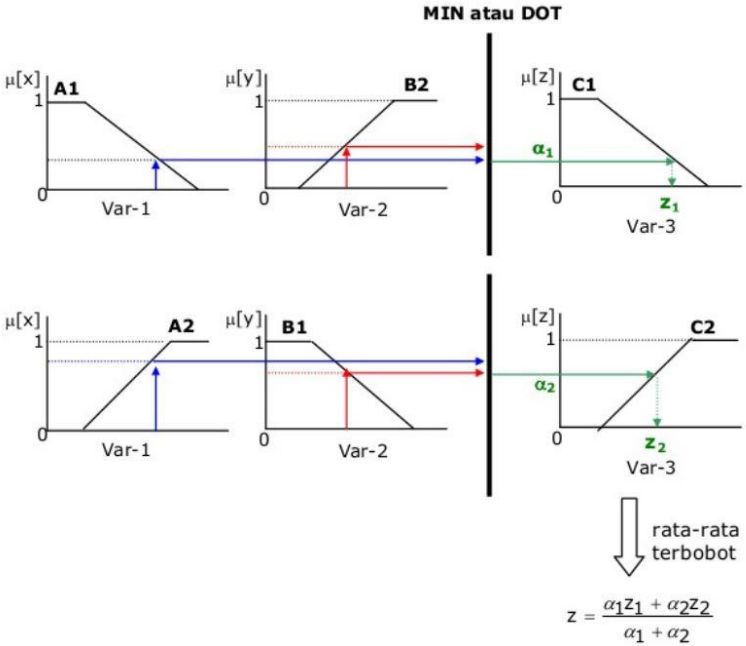
c. Aturan – aturan (rules)
Aturan-aturan yang akan dijadikan dasar untuk mencari nilai dari crisp output yang akan dihasilkan.

d. Defuzzifikasi
Merupakan proses merubah kembali data yang dijadikan fuzzy ke dalam bentuk crisp kembali.

e. Nilai output
Merupakan hasil akhir yang dapat dipakai untuk pengambilan keputusan. Namun terkadang sistem fuzzy dapat berjalan tanpa harus melalui komposisi atau dekomposisi fuzzy. Nilai output dapat diestimasi secara langsung dari nilai keanggotaan yang berhubungan dengan antesedennya.

2.1.7 Metode Tsukamoto

Dalam metode Tsukamoto, tiap konsekuen pada aturan yang berbentuk *if-then* harus direpresentasikan dengan suatu himpunan fuzzy dengan fungsi keanggotaan yang monoton. Hasilnya, output inferensi yang dihasilkan dari tiap-tiap aturan diberikan secara tegas(*crisp)* berdasar $\propto$-predikat. Hasil akhirnya diperoleh dengan rata-rata terbobot[2].



Gambar 2.1.4 Inferensi Fuzzy dengan metode Tsukamoto[2]

2.2 OpenCV

OpenCV memiliki singkatan *Open Source Computer Vision*. OpenCV merupakan sebuah *software library* bebas (*open source*) yang digunakan untuk operasi *computer vision* dan *machine learning*. Pustaka ini termasuk pustaka yang *opensource*. OpenCV pertama kali diluncurkan tahun 1999 oleh Intel sebagai lanjutan proyek aplikasi berbasis CPU.

Library ini ditulis dalam bahasa C dan C++ dan dapat dijalankan pada sistem operasi Linux,Windows dan Mac OS X. Selain itu terdapat pengembangan pada *Python*,*Ruby*,*Matlab*, dan bahasa pemrograman lain. OpenCV telah dibangun untuk menyediakan sebuah infrastruktur umum untuk beberapa aplikasi *computer vision* dan untuk mempercepat penggunaan dari mesin persepsi dalam produk komersial. OpenCV mempermudah bisnis-bisnis untuk memanfaatkan dan memodifikasi kode. *Library* OpenCV mempunyai lebih dari 2500 algoritma yang telah dioptimalkan.

Gambar 2.2 Logo OpenCV[4]

2.3 Kamera Omnidireksional

Dalam fotografi omnidireksional kamera adalah sebuah kamera yang memiliki sudaut pandang 360 derajat pada area horizontal atau sudut pandang yang melingkupi hampir seluruh bola. Kamera omnidireksional[11]

digunakan pada fungsi yang memerlukan sudut pandang luas seperti pada fotografi panorama dan beberapa keperluan robotika.

Dalam robotika, kamera omnidireksional sering digunakan untuk odometri secara visual dan untuk menyelesaikan *simultaneous localization and mapping(SLAM).* Karena kemampuan kamera omnidireksional untuk mengambil gambar dengan sudut pandang 360 derajat, hasil yang lebih baik akan didapatkan untuk *optical flow* dan *feature selection and matching.*

Gambar 2.3 Kamera Omnidireksional

2.4 Regresi Polinomial

Metode regresi adalah suatu metode statistik untuk mnyelidiki dan memodelkan hubungan antara variabel respon *Y* dan variabel prediktor *X*. Misal diberikan himpunan data {(Xi,Yi)},i = 1, ...,n. Secara umum hubungan antara Y dan X dapat ditulis sebagai berikut:

$$Y_i = m(X_i) + \varepsilon_i$$

Dengan $m(X_i)$ adalah suatu fungsi regresi yang belum diketahui dan ingin ditaksir, dan ε_i adalah suatu variabel acak yang menggambarkan variasi Y di sekitar $m(X_i)$.

2.5 Open Framework

OpenFramework adalah sebuah perangkat *open source C++* yang didesain untuk mendukung proses kreatif dengan menyediakan bingkai atau lembar kerja yang sederhana dan intuitif untuk bereksperimen [5]. openFramework dapat berjalan di lima sistem operasi yaitu Windows, OSX, Linux, iOS, dan Android. Selain itu openFramework dapat dijalankan di empat IDEs yaitu Xcode, Codeblock, Visual Studio dan Eclipse.

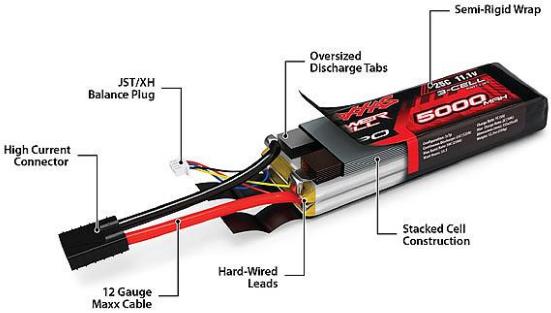
OpenFramework didistribusikan dibawah lisensi oleh MIT. Hal ini membebaskan setiap orang untuk menggunakan openFramework baik untuk tujuan komersial ataupun non-komersial, Umum ataupun privat, *open source* atupun *closed source.*

Gambar 2.5 Logo openFrameworks[5]

2.6 Baterai Lipo

Komponen sumber energi utama dari sebuah *quadcopter* adalah baterai LiPo.Seperti gambar 2.9, baterai lipo tidak menggunakan cairan sebagai elektrolit melainkan menggunakan elektrolit polimer kering

yang berbentuk seperti lapisan plastik film tipis. Lapisan film ini disusun berlapis-lapis diantara anoda dan katoda yang mengakibatkan pertukaran ion. Dengan metode ini baterai LiPo dapat dibuat dalam berbagai bentuk dan ukuran. Diluar dari kelebihan arsitektur baterai LiPo, terdapat juga kekurangan yaitu lemahnya aliran pertukaran ion yang terjadi melalui elektrolit polimer kering. Hal ini menyebabkan penurunan pada *charging* dan *discharging rate.*



Gambar 2.6 Konfigurasi Baterai Lipo

2.7 STM32f4 Discovery

STM32F4 merupakan mikrokontroler berbasis ARM 32 bit. Mikrokontroler jenis STM32F4 menggunakan ARM core-M4F yang dapat melakukan pengolahan sinyal digital. STM32F4 memiliki fitur penambahan kecepatan clock yang lebih tinggi dari STM32F2, 64K CCM static RAM, full duplex I^2S, dan memiliki kecepatan konversi ADC. Gambar 2.7 merupakan gambar bagian dari STM32F4 Discovery. Board developer ini sudah menyediakan downloader ST-LINK sehingga cukup diperlukan koneksi USB untuk melakukan proses download program.

Gambar 2.7 STM32F4 - Discovery

2.8 Rotary Encoder

Rotary encoder adalah sensor penghitung banyak putaran yang dilakukan pada suatu benda yang berputar. Pada rotary encoder terdapat sebuah piringan yang memiliki bagian berlubang dan tidak berlubang seperti gambar 2.8. ketika sensor terkena daerah berlubang maka photo sensor mendeteksi cahaya dari LED, jika cahaya LED tertutup daerah yang tidak berlubang maka photo sensor tidak akan mendeteksi cahaya dari LED. Perbedaan sinyal ini akan dideteksi mikrokontroller dan akan melakukan suatu perhitungan untuk diproses selanjutnya.

Gambar 2.8 Rotary Encoder

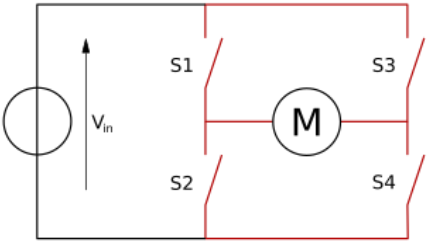
2.9 Mini PC

Mini PC adalah sebuah computer kecil yang memiliki kemampuan yang sama seperti laptop atau PC *desktop*. Mini PC yang kita gunakan di project akhir ini adalah Intel NUC yang memiliki spec cukup tinggi untuk pengolahan citra dan algoritma yang kita gunakan sekarang. Gambar 2.9 adalah salah satu contoh mini pc yang kita gunakan.

Gambar 2.9 Mini PC

2.10 Driver Motor

Driver Motor adalah rangkaian pengendali motor. Rangkaian ini digunakan untuk mengendalikan motor dari divais ber arus kecil yang tidak bias mengendalikan motor yang ber arus besar. Rangkaian *diver* motor yang sering dipakai adalah rangkaian konfigurasi jembatan H atau biasa disebut *H-Bridge* seperti gambar 2.10. Rangkaian ini dapat mengatur arah putaran motor. Jika saklar S1 dan saklar S4 disambungkan dan saklar S2 dan saklar S3 tidak disambungkan, arus akan mengalir dari tegangan sumber ke sisi kiri motor. Hal ini akan membuat gerakan motor searah jarum jam. Sebaliknya, jika saklar yang tersambung adalah saklar S2 dan saklar S3, arus akan mengalir terbalik dan menyebabkan arah motor berbalik juga.



Gambar 2.10 Rangkaian H-Bridge

2.11 Motor DC

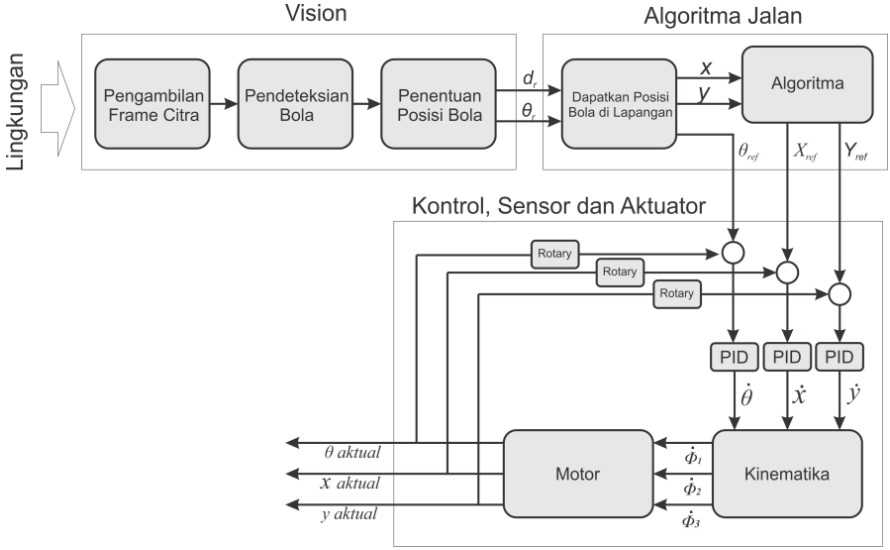
Motor DC merupakan perangkat elektromagnetik yang berfungsi untuk mengubah energi listrik menjadi energi mekanik. Prinsip kerja Motor DC memanfatkan hukum Lorentz. Ketika ada arus yang melewati rotor, interaksi antara magnet yang berada pada motor dan medan magnet yang ditimbulkan oleh arus di rotor menyebabkan rotor berputar.[12]

Gambar 2.11 Motor DC[12]

BAB 3
PERANCANGAN SISTEM

3.1 Perancangan Sistem

Pada pernacangan robot penjaga gawang kali ini ada beberapa bagian penting yang saling berkaitan. Sistem ini terrepresentasikan dalam gambar blok diagram pada gambar 3.1. pada blok tersebut, penjelasan detail tentang tiap blok diagram akan dijelaskan pada sub-bab 3.1.1-3.1.3.

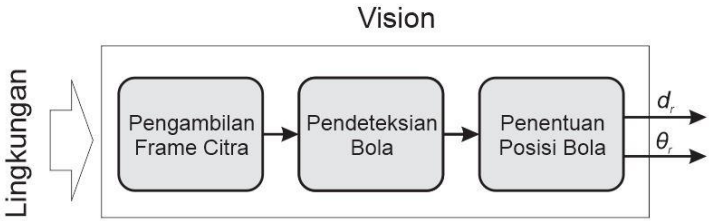


Gambar 3.1 Blok diagram sistem

3.1.1 Vision

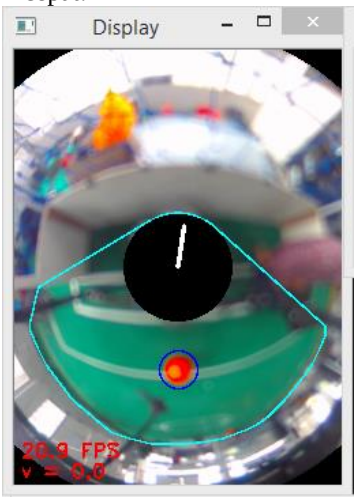
Blok vision adalah hal utama dalam pengambilan data bola dan lapangan sepak bola lalu diolah agar didapatkan informasi posisi bola terhadap robot. posisi bola terhadap robot dinyatakan dalam koordinat polar yaitu jarak bola terhadap robot yang dinyatakan dalam d_r dan sudut bola terhadap arah depan robot yang dinyatakan dalam θ_r dengan memosikan robot

sebagai titik origin dari koordinat tersebut. Blok diagram vision lebih jelas dapat ditunjukkan pada Gambar 3.2.



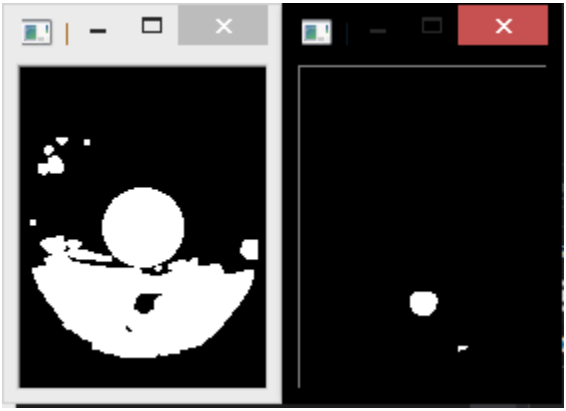
Gambar 3.1.1 Blok Diagram Vision

Pada proses yang pertama yaitu proses pengambilan *frame* citra. Proses ini adalah proses pertama yang harus dilakukan pada bagian *in proccess*. Citra gambar di tetapkan dengan resolusi 640 x 480. Semakin tinggi resolusi akan semakin jelas informasi dari citra yang nampak, namun memiliki kelemahan yaitu waktu pemrosesan yang relatif lebih lama. Sedangkan jika menggunakan resolusi gambar yang terlalu kecil akan semakin banyak informasi dari citra yang hilang meskipun waktu pemrosesan relatif lebih cepat.



Gambar 3.1.2 Hasil Pendeteksian Bola

Pada proses pengambilan citra, citra diambil dalam bentuk citra RGB atau BGR. Kedalaman warna yang dipilih untuk setiap kanalnya adalah sebesar 8 bit sehingga setiap kanal dapat merepresetasikan 255 tingkat kecerahan yang berbeda. Dengan adanya 3 kanal yaitu *red*, *green*, dan *blue*, kombinasi atau variasi warna yang dapat dihasilkan adalah sebesar 255^3 atau sebesar 16.581.375 warna. informasi warna dalam bidang RGB(*red, green, blue*) akan disusun ulang pada bidang HSV(*hue, saturation, value*). Hal ini perlu dilakukan mengingat informasi yang disajikan oleh format RGB tidak memisahkan antara informasi warna dan informasi kecerahan. Sedangkan pada format HSV informasi tersebut dipisahkan oleh sebuah kanal *value.*



Gambar 3.1.3 Hasil Threshold Lapangan (kiri) dan Bola (kanan)

Pada tahap selanjutnya dilakukan proses pencarian posisi bola dari citra yang telah di proses sebelumnya. Pendeteksian bola dilakukan dengan memanfaatkan karakteristik warna bola yaitu warna oranye. Untuk mengurangi noise akibat salah deteksi warna oranye lain, pencarian warna oranye bola hanya dilakukan di atas lapangan berwarna hijau. Sehingga jika terdapat warna oranye diatas warna bukan warna hijau lapangan maka bisa dipastikan bahwa objek tersebut bukanlah bola. Oleh karena itu diperlukan juga proses pendeteksian lapangan.

Setelah bola dapat diteksi, langkah selanjutnya adalah menentukan dimana posisi bola tersebut pada citra. Pada tahap ini proses penentuan posisi bola adalah dengan merubah koordinat bola pada citra ke

koordinat sudut dan jari-jari terhadap titik tengah citra. Proses ini dilakukan mengingat kamera yang digunakan adalah kamera omni direksionnal.

Saat terdapat bola terdeteksi, posisi bola pada citra akan diketahui juga dengan melihat titik pusat bola pada citra. Diasumsikan posisi bola dari proses pendeteksian bola dinyatakan dengan x_b dan y_b. Dan posisi tengah citra diasumsikan dengan x_c dan y_c. Untuk menkonversi posisi tersebut menjadi jarak pusat bola terhadap titik tengah citra (r_p) dan sudut bola terhadap titik tengah (θ) citra digunakan persamaan berikut:

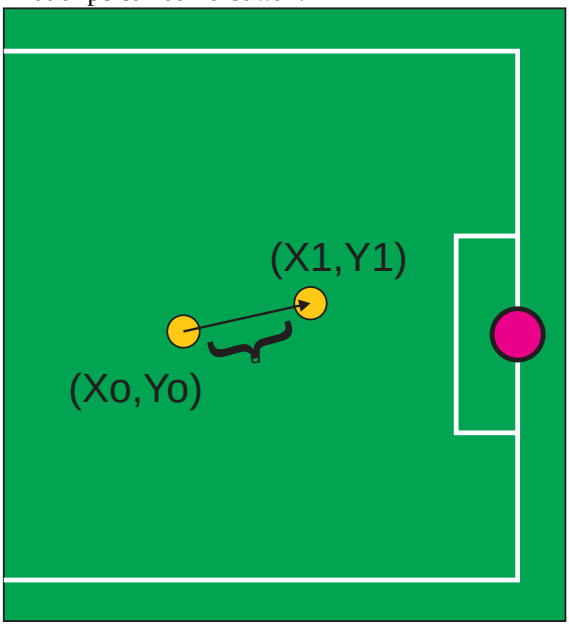
$$r_p = \sqrt{(x_b - x_c)^2 + (y_b - y_c)^2}$$

$$\theta = \tan^{-1} \frac{(y_o - y_c)}{-(x_o - x_c)}$$

Pada tahap sebelumnya didapatkan informasi tentang posisi bola terhadap titik tengah pada citra. Namun jari-jari yang diperoleh pada informasi ini tidak merepresentasikan nilai jarak sesungguhnya. Hubungan antara jari-jari bola terhadap titik tengah pada citra dan jarak bola sesungguhnya terhadap robot juga tidak linear. Oleh karenanya diperlukan sebuah proses untuk menentukan posisi bola terhadap robot dari informasi yang ada. Salah satu pendekatan yang dilakukan yaitu dengan metode regresi polinomial.

Pada tahap selanjutnya kita akan mencari kecepatan bola. Kecepatan bola disini kita cari dengan cara *sampling* perpindahan bola tiap 150 ms, perpindahan ini terdari 2 sumbu, yaitu sumbu x dan sumbu y. titik awal bola kita dapat sebut sebagai x_o dan y_o, titik akhir bola saat sampling selanjutnya dapat kita sebut x_1 dan y_1. Setelah titik tersebut didapatkan kita selanjutnya mencari selisih dari kedua titik tersebut. Selisih dari kedua titik tersebut dapat kita sebut sebagai vektor kecepatan tiap sumbu. Setelah mendapatkan vektor kecepatan tiap sumbu, selanjutnya akan didapatkan resultan dari kecepatan denga metode *phytagoras*. Data resultan ini kita gunakan sebagai data kecepatan bola yang digunakan sebagai input *fuzzy,* lalu ketika penjaga gawang merespon bola dan bergerak sesuai *output* fuzzy (z)yang bergerak hanya terhadap sumbu x lapangan maka kecepatan relative bola akan berbeda dari yang seharusnya, maka perlu ada pertambahan di vector kecepatan x yang ditambah dengan kecepatan gerak robot

terhadap sumbu-x. Alur persamaan dengan *pseudocode* dan ilustrasinya dapat kita lihat di persamaan dibawah.

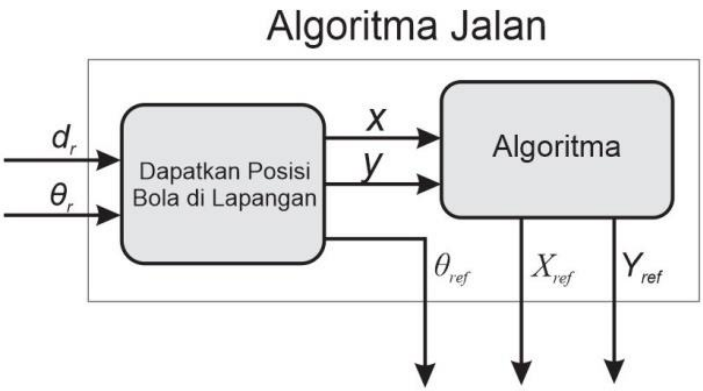


Gambar 3.1.4 Ilustrasi Mendapatkan Kecepatan Bola

time = ofGetSystemTime();
If(time – timeprev < 150)
{
$Vx = x_1 - x_0 + z$
$Vy = y_1 - y_0$
$Vresultan = \sqrt{Vx^2 + Vy^2}$
$if\,(Vy < 0)$
{
Vresultan = -Versultan
}
else Vresultan = Versultan
timprev = time
$x_1 = x_0$
$y_1 = y_0$
}

3.2 Algoritma Jalan

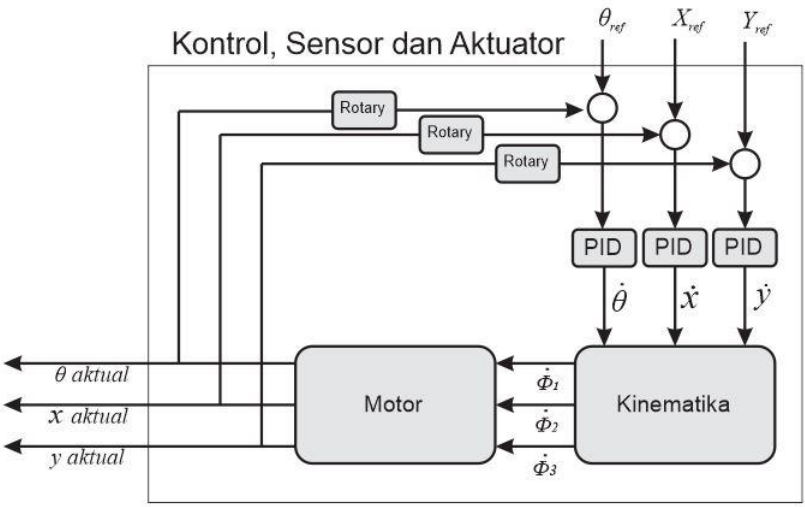
Algoritma jalan atau gerak robot di gunakan untuk membuat rencana pergerakan robot yang efektif untuk menuju sebuah titik tertentu. Pergerakan yang dilakukan penjaga gawang tentunya beda dengan pemain yang seharusnya lebih mudah karena hanya memerlukan gerakan terhadap sumbu *x* saja. Rencana algoritma jalan yang dilakukan adalah dengan penghadangan bola meggunakan metode Fuzzy yang kita bahas dalam tugas akhir kali ini. Diagram blok dari sistem ini ditunjukkan pada gambar 3.2.1



Gambar 3.2.1 Blok Diagram Algoritma Perencanaan Gerak

3.3 Kontrol Pergerakan

Bagian ini berfungsi untuk menggerakkan robot menuju posisi yang diinginkan sesuai keluaran dari proses sebelumnya. Kontrol PID digunakna untuk mendapatkan posisi yang diinginkan. *Feedback* posisi robot berada menggunakan sistem odometry dengan rotary encoder yang dipasang di bagian bawah robot. Selain itu arah dari robot bisa diperoleh dengan menggunakan sensor IMU MPU 6050. Blok kontrol pergerakan lebih jelasnya ditunjukkan pada Gambar 3.3.1.



Gambar 3.3.1 Blok Kontrol Robot

3.3.1 Perancangan Sistem Gerak

Robot di desain dengan menggunakan penggerak 4 WD dengan menggunakan motor penggerak utama PG-45 (dapat dilihat pada Gambar 3.3.2) dengan tegangan kerja sebesar 24 volt dan kecepatan putar 400rpm. Motor ini telah dilengakapi dengan *gearbox planetary* dengan perbandingan gear 1:19. Sebagai driver motor untuk menentukan kecepatan dan arah putar motor digunakan modul driver motor BTN 7970.

Robot di desain menggunakan sistem penggerak roda omni dengan jarak antar roda berjarak 110 derajat dan 60 derajat. Roda omni yang digunakan pada robot memiliki berdiameter 10 cm dan terbuat dari material aluminium. Untuk lebih jelasnya roda omni yang digunakan pada robot ditunjukkan pada Gambar 3.3.3

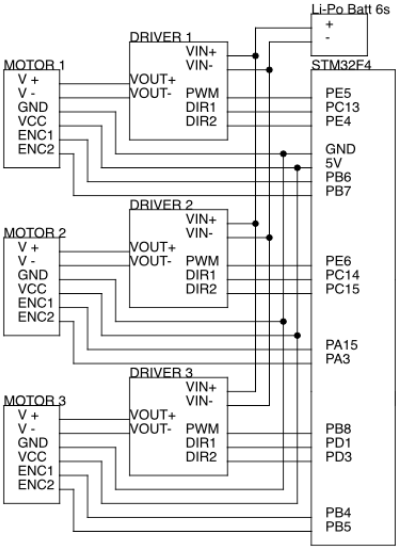
Gambar 3.3.2 Motor DC PG-45

Gambar 3.3.3 Roda Omni

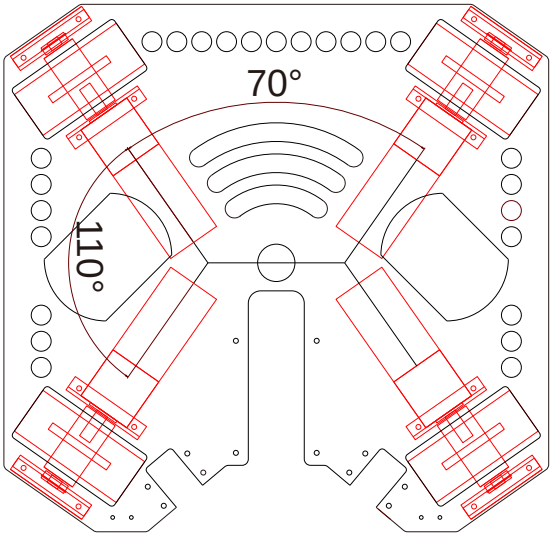
Dengan desain penempatan motor seperti pada Gambar 3.3.5, akan diperoleh kinematika pergerakan robot sebagai berikut:

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \\ \dot{\phi}_4 \end{bmatrix} = \begin{bmatrix} sin(\theta) & -cos(\theta) & R \\ -sin(\theta+\alpha_2) & -cos(\theta+\alpha_2) & R \\ -sin(\theta+\alpha_3) & cos(\theta+\alpha_3) & R \\ sin(\theta+\alpha_4) & cos(\theta+\alpha_4) & R \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Interkoneksi antar motor penggerak, driver motor, dan mikrokontroller STM32f4 ditunjukkan pada Gambar 3.3.4.



Gambar **3.3.4** Interkoneksi Motor, Driver Motor dan STM32F4



Gambar 3.3.5 Base Robot

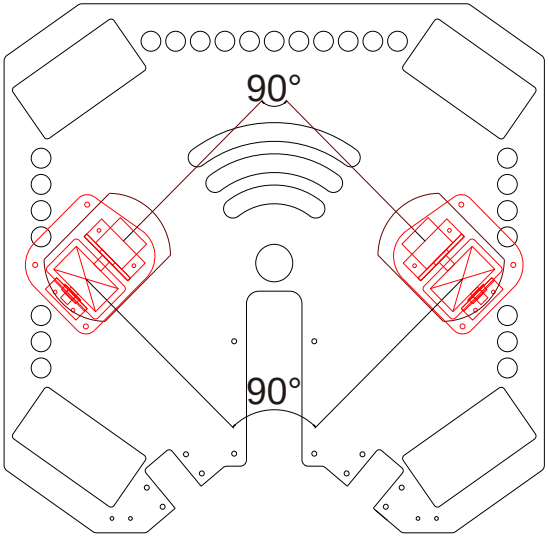
3.3.2 Perancangan Sistem Odometri

Sistem odometri digunakan untuk mendapatkan posisi robot dilapangan dan dijadikan feedback untuk kontrol posisi. Sistem odometri dibentuk dengan menggunakan 2 rotary encoder yang dipasang pada bagian bawah robot. Rotary encoder yang digunakan pada robot dapat dilihat pada Gambar 2.8.

Desain base untuk rotary encoder dapat dilihat pada Gambar 3.3.6. Peletakan rotary encoder satu dengan yang lainnya berjarak 90 derajat. Dengan desain tersebut pergerakan dari robot akan dapat dinyatakan dengan persamaan sebagai berikut:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -cos(225° + \theta) & -cos(135° + \theta) \\ -sin(225° + \theta) & -sin(135° + \theta) \end{bmatrix}$$

Sudut dari robot diperoleh melaui sensor IMU MPU 6050 dan dapat dilihat pada Gambar 3.5.6]

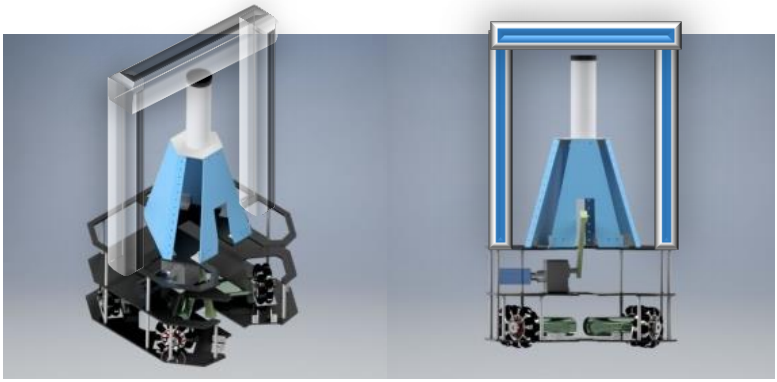


Gambar 3.3.6 Desain base untuk rotary encoder

3.4 Desain mekanik robot

Desain Mekanik robothampir keseluruhan terbuat dari pelat aluminium, ada sebagian kecil bagian robot yang juga terbuat dari akrilik Desain dari *base body* robot ditunjukkan pada Gambar 3.3.5.

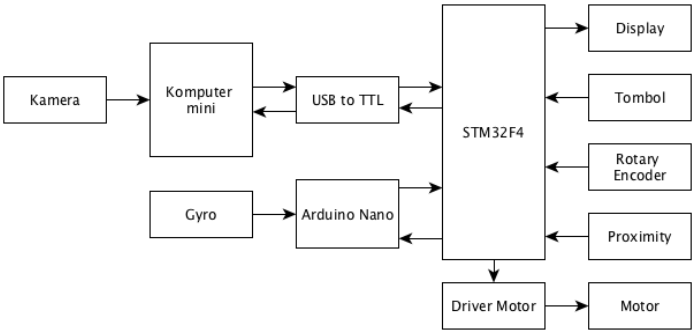
Desai dari robot penjaga gawang ini sama seperti robot pemain, hanya bagian muka yang berbeda dari pemain. Bentuk dari robot penjaga gawang adalah kotak dengan bagian atas mengerucut. Dan juga ada tabung kamera ditempat paling atas. Bagian atas robot terbuat dari aluminium *hollow* yang berfungsi sebagai tangan robot. Panjang-lebar robot ini yaitu 45 cm x 45 cm. Desain 3D dari robot dapat dilihat pada Gambar 3.4.1.



Gambar 3.4.1 Gambar desain 3D Robot

3.5 Desain elektronik robot

Desain elektronik dari robot ditunjukkan pada Gambar 3.5.1. Sistem elektronik yang ditunjukkan pada gambar tersebut adalah sistem elektronik keseluruhan dari platfom robot yang dibuat sendiri. Namun pada tugas akhir kali ini fokus pengerjaannya pada bagian kamera dan komputer mini.



Gambar 3.5.1 Blok diagram sistem elektronik robot

3.5.1 Supply

Supply utama dari robot ini adalah baterai LiPo(Lithium Polymer) yang sebesar 24 volt. Ada dua baterai yang mensuplai system kerja robot. Baterai pertama digunakan untuk menyuplai energy ke PC dan Mikrokontroller. Baterai yang lain digunakan untuk mensuplai motor dan aktuator lain. Untuk sumber baterai pertama akan masuk ke modul buck converter untuk PC dan dua buck converter untuk mikrokontroler dan sensor yang digunakan terpisah. Pada buck converter pada PC dapat dilihat di gambar 3.5.9.1. pada buck ini akan menyalurkan tegangan 19,5 volt untuk catu daya PC utama. umtuk buck converter pada mikro dan sensor ada dua buah buck, yang satu mensuplai mikrokontroler sebesar 5 volt dan yang satunya mensuplai sensor sebesar 12 volt.

Gambar 3.5.1.1 Baterai Lipo

3.5.2 Mini PC Intel Nuc

Pemrosesan citra dan algoritma dalam tugas akhir ini diproses didalam sebuah mini pc kecil bernama Intel NUC. Mini pc ini membutuhkan performa yang memadai, mengingat pemrosesan yang dilakukan butuh kecepatan pemrosesan yang tinggi maka kita membutuhkan mini pc yang handal di kelasnya.

Spesifikasi dasar dari Intel NUC yang digunakan adalah sebagai berikut:

- Prosesor Intel® Core™ i7-6770HQ (6M Cache, hingga 3.50 GHz)
- RAM 8GB DDR4
- M2 SSD 120GB
- TDP 45W
- Input DC 19V
- 4 PORT USB
- HDMI

Gambar 3.5.2.1 Mini PC Intel NUC5I7KYK

3.5.3 STM32F4-discovery

Pada tugas akhir ini STM32F4-discovery adalah mikrokontroler yamg berperan sebagai *slave.* Data dari PC akan diterima oleh mikrokontroler dan akan dilanjutkan ke suatu respon tertentu sesuai perintah dari PC seperti penggerak motor dan aktuator lain. Di mikrokontroler ini juga mengirim data berupa data sensor, tombol, posisi robot sekarang dan lain-lain ke PC. Komunikasi yang dilakukan menggunakan metode UART(Universal Asyncronous Recieve Transmit), sehingga dapan mengirim dan menerima data.

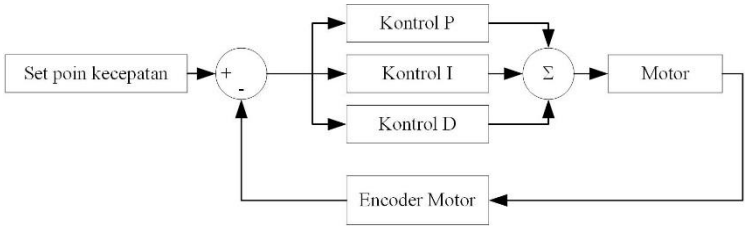
Salah satu data yang diproses adalah data pergerakan motor. PC mengirimkan data berupa kecepatan arah sumbu x, kecepatan arah sumbu y, kecepatan rotasi, arah depan robot. Dari data yang diperoleh

tersebut STM32F4-discovery mengolah nilai-nilai tersebut menggunakan invers kinematik sesuai dengan persamaan berikut:

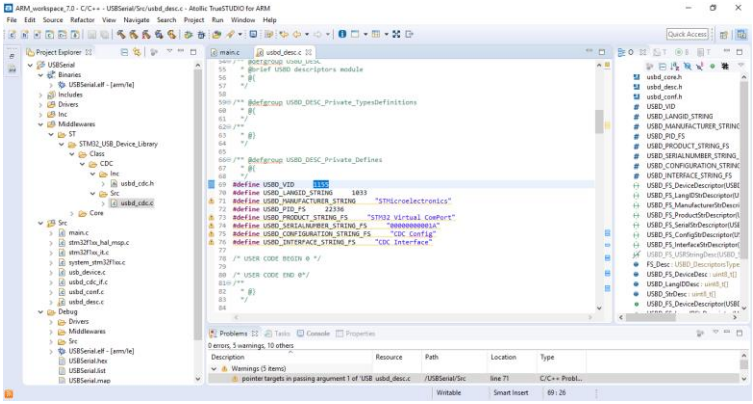
$$Ref_1 = K(C\omega - v_x sin30° + v_y cos55°)$$
$$Ref_2 = K(C\omega - v_x sin30° - v_y cos55°)$$
$$Ref_3 = K(C\omega + v_x sin30° - v_y cos55°)$$
$$Ref_3 = K(C\omega + v_x sin30° + v_y cos55°)$$

Dari persamaan tersebut didapatkan kecepatan kecepatan-kecepatan yang harus diberikan kepada tiap motor.

Didalam STM32F4-discovery ada proses kontrol kecepatan motor dengan feedback berasal dari putaran sensor encoder motor. Kecepatan motor didapatkan dengan memanfaatkan peripheral *quadrature encoder* menggunakan *Timer dan Counter* STM32F4-discovery. Prinsip kerjanya yaitu dengan mengambil nilai *tick* saat berputar dan kemudian di reset dengan periode tertentu. Kontrol kecepatan yang dilakukan oleh STM32F4-discovery menggunakan PID dengan diagram blok ditunjukkan pada Gambar 3.5.3.1.

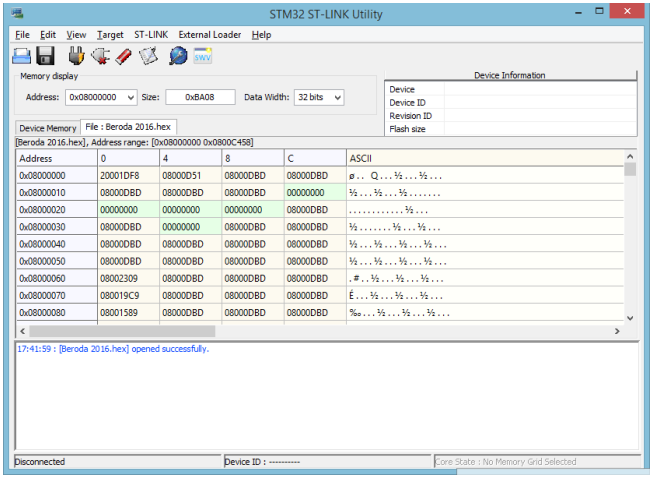


Gambar 3.5.3.1 Diagram kontrol PID pada motor



Gambar 3.5.3.2 Tampilan IDE Atollic

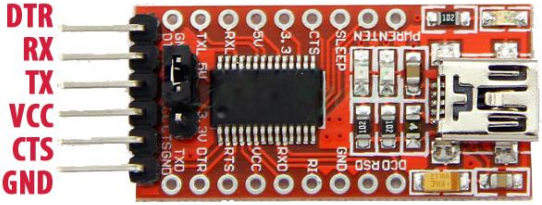
Proses pemrograman STM32F4-discovery menggunakan software IDE Atollic. Compiler yang digunakan adalah "GNU Tools for ARM Embedded Processor". Kompiler ini membantu IDE Atollic untuk menghasilkan file hex dari kode yang telah dibuat. Proses mendownload program dilakukan dengan menggunakan kabel USB to mini-USB. Selain itu proses download juga dibantu oleh software ST-LINK untuk membuka file hex dan mendownloadnya ke chip mikrokontroler.



Gambar 3.5.3.3 Tampilan Software ST-LINK

3.5.4 USB to Serial Converter

Perangkat USB to Serial Converter digunakan sebagai alat bantu komunikasi antara mikrokontroller STM32F4-discovery dan mini PC. Melalui alat ini data-data dari STM32F4-discovery dapat dikirimkan ke Mini PC dan begitu pula sebaliknya. Cara kerja alat ini adalah dengan merubah data USB menjadi level tegangan TTL begitu pula sebaliknya. Seri USB to Serial Converter yang dipakai pada TA kali ini adalah FTDI FT232R. Penggunaan alat ini adalah dengan menghubungkan alat tersebut dengan komputer melalui USB dan menhubungkan device melaui pin TX RX dengan device lain seperti mikrokontroler atau sejenisnya.



Gambar 3.5.4.1 USB to Serial Converter

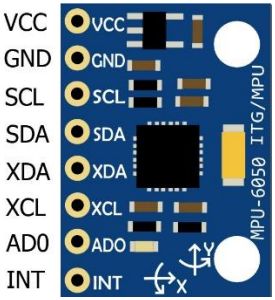
3.5.5 Arduino Nano

Pada tugas akhir kali ini arduino nano digunakan sebagai pengakses dan pengolah data dari sensor IMU MPU6050. Setelah selesai diolah data tersebut kemudian dikirimkan ke STM32F4-discovery. Data yang dikirimkan hanyalah data rotasi robot pada sumbu z. Karena data rotasi pada sumbu yang lain tidak digunakan pada robot ini. Data rotasi tersebut selanjutnya akan digunakan pada proses perhitungan odometri dan penentuan orientasi robot.

Gambar 3.5.5.1 Arduino nano

3.5.6 Sensor IMU(Inertia Measurement Unit)

Sensor IMU yang digunakan pada tugas akhir kli ini adalah MPU 6050. Sensor ini adalah sensor MEMS(*Micro Electro Mechanical System*). Sensor ini berkomuniasi dengan perangkat lain menggunakan interface I2C. Sensor ini adalah salah satu sensor cerdas karena dapat menyediakan data yang telah diolah melalui sebuah pemrosesan internal yang lebih dikenal dengan DMP(*Digital Motion Processor*). Sensor ini mengkombinasikan antara 3 axis gyroskop dan 3 axis akselerometer dalam sebuah chip kecil.



Gambar 3.5.6.1 MPU-6050 dan pin outnya[6]

3.5.7 Sensor garis

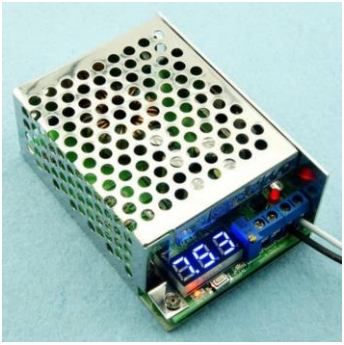
Sensor garis pada sistem robot digunakan sebagai alat kalibrasi posisi robot terhadap lapangan memanfaatkan garis-garis yang disediakan. Sensor garis yang digunakan pada robot kali ini adalah sensor autonic tipe BF5R. Output dari sensor ini dihubungkan dengan pin logic pada STM32F4 yang diberi pull up internal sehingga mendeteksi logic 0 saat mendeteksi garis dan logic 1 saat tidak mendeteksi garis. Konfigurasi dari sensor garis ini adalah GND dihubungkan dengan GND supply, VCC pada sensor dihubungkan dengan supply 12 V, sedangkan out pada sensor dihubungkan dengan pin logic pada STM32F4.

Gambar 3.5.7.1 Modul Sensor Garis

3.5.8 Buck Converter untuk Mini PC, Mikrokontroler dan Sensor

Ada dua Buck converter ini digunakan sebagai reguulator tegangan dari sumber utama yaitu baterai lipo 6 Sel. Mini PC intel Nuc yang digunakan pada tugas akhir kali ini memiliki tegangan kerja 19V DC. Dan sumber mikrokontroler dan sensor adalah 5V DC dan 12 V DC. Oleh karena itu diperlukan sebuah regulator untu menurunkan tegangan penuh baterai Lipo 6 Sel yaitu 25.2 V menjadi 19V, 12V dan 5V. Tagangan input dari buck converter ini adalah 10 – 45V. Arus maksimal yang mampu dihasilkan dari buck converter ini adalah sebesar 10 A.

Gambar 3.5.8.1 Modul Buck Converter 12V dan 5V



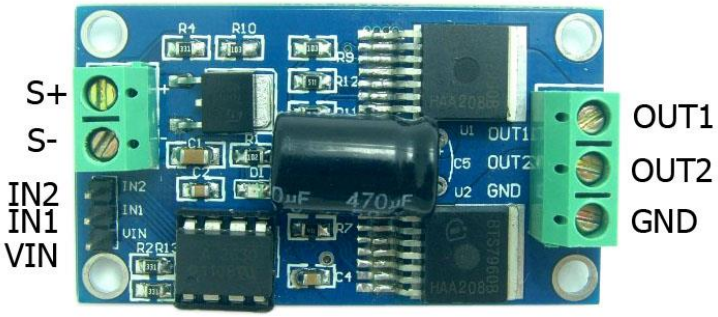
Gambar 3.5.8.2 Modul Buck Converter 19V

3.5.9 Driver Motor

Driver motor digunakan untuk mengaktifkan motor serta menentukan arah putar motor sesuai sinyal diberikan oleh mikro kontroler. Driver motor yang digunakan pada tugas akhir kali ini adalah module driver motor BTN7970 sebanyak 3 buah. Modul driver motor ini terdiri dari 2 IC *half-bridge* BTN7970 sehingga memungkinkan untuk melakukan kontrol putaran dua arah baik serah jarum jam ataupun berlawanan arah jarum jam. Driver motor ini memiliki kemampuan mengalirkan arus hingga 50A.

Modul driver motor ini memiliki 5 input yang terdiri dari 3 pin kontrol (VIN, IN1, dan IN2) dan 2 pin supply (+ dan -). Selain itu modul ini memiliki 3 output yang ber label GND, OUT1, dan OUT2.

Modul ini memeliki kemampuan untuk men-*drive* 2 motor sekaligus dengan menyambungkan input motor masing-masing ke output OUT1 – GND dan OUT2 – GND. Kontol dilakukan melaui pin VIN sebagai enable serta IN1 dan IN2 sebagai kontrol kecepatan biasanya menggunakan PWM. Namun jika itu dilakukan setiap motor hanya dapat berputar ke satu arah karena masing-masing motor hanya di *drive* oleh satu IC half bridge. Untuk melakukan kontrol full dua arah satu motor harus dihubungkan dengan output OUT1-OUT2. Kontrol kecepatan dan arah putar motor dilakukan melalui pin kontrol IN1 dan IN2. Jika IN 1 berlogika 1 dan IN2 berlogika 0 maka arahh putar motor adalah berlawanan arah jarum jam sedangkan jika input IN1 berlogika 0 dan input IN2 berlogika 1 maka arah putaran motor adalah serah dengan jarum jam. Selain itu input VIN digunakan sebagai kontrol kecepatan



Gambar 3.5.9.1 Modul Driver Motor BTN 7970

menggunakan PWM. Jika PWM memiliki duty cycle rendah maka kecepatan motor relatif pelan sedangakan jika PWM memiliki duty cycle tinggi kecepatan motor relatif tinggi.

3.5.10 Rotary encoder

Rotary encoder yang digunakan pada tugas akhir kali ini adalah autonic seri E30. rotary encoder digunakan sebagai komponen untuk odometry robot yang akan menjadi *feedback* posisi robot. Rotary encoder yang digunakan pada tugas akhir ini memiliki spesifikasi tegangan kerja 5 - 24V DC ± 5% dan 200 pulsa per rotasi.

Terdapat 4 kabel yang menjuntai dari sensor tersebut masing-masing adalah supply +(VCC), supply –(GND), A, dan B. Supply + dan supply – adalah jalur supply untuk sensor yang pada desain kali ini dihubungkan langsung pada baterai lipo 3 sel. Sedangkan A dan B adalah output dari sensor yang dihubungkan pada counter melalui pin di STM32F4 dimana pulsa dari kedua output itu berbeda fasa sebesar 90 derajat. Perbedaan fasa ini digunakan sebagai tanda arah putaran dari motor teresebut apakah berlawanan ataukah serah dengan arah putaran jarum jam.

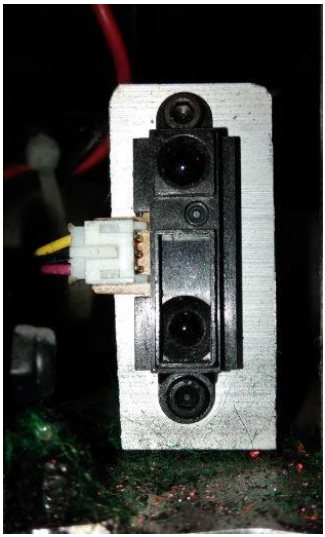
Gambar 3.5.10.1 *Rotary Encoder*

Gambar 3.5.10.2 *Wire out* dari rotary encoder

3.5.11 Sensor IR Range Finder

Sensor IR Range Finder digunakan untuk mendeteksi ada atau tidaknya objek. Sensor ini bekerja dengan tegangan 5V. Sensor ini akan memancarkan sinar inframerah yang pantulannya akan dideteksi oleh suatu photodiode khusus. Intensitas cahaya pantulan tersebut akan berubah menjadi suatu nilai analog yang akan keluar sebagai output sensor. Namun data tersebut tidak linier. Maka perlu dilinierkan menggunakan suatu persamaan sesuai yang ada di datasheet. Konfigurasi dari sensor ini adalah VCC dihubungkan pada supply 5V dan GND dihubungkan pada supply GND. Sedangkan out pada sensor dihubungkan dengan pin ADC pada STM32F4 untuk membaca output analog dari sennsor tersebut.

Pada tugas akhir kali ini sensor IR Range Finder digunakan untuk mendeteksi ada atau tidaknya bola pada penggiring bola. Informasi ini akan digunakan lebih lanjut oleh robot untuk mengaktifkan penggiring, membawa bola ke gawang lawan, melakukan proses umpan dan lain sebagainya.



Gambar 3.5.11.1 Sensor IR Range Finder

Gambar 3.5.11.2 *Wire Out* Sensor Range Finder

3.6 Perencanaan Fuzzy Logic

Pada subbab ini akan membahas komponen apa saja ang diperlukan untuk menjalankan algoritma fuzzy logic. Perencanaan ini meliputi fungsi keanggotaan tiap variable input, perencanaan rule dan perencanaan keluaran fuzzy yang diinginkan.

3.6.1 Perancangan Fungsi Keangotaan

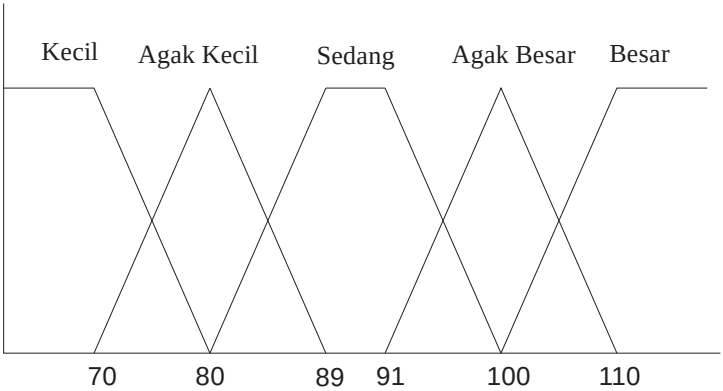
Dalam Tugas Akhir ini, setiap variabel fuzzy menggunakan fungsi keanggotaan bahu, linier turun, linier naik, trapezoid dan segitiga sebagai pendekatan untuk memperoleh derajat keanggotaan suatu nilai dalam suatu himpunan fuzzy.

Bentuk kurva berikut ini adalah kurva default dari beberapa faktor yang mempengarui himpunan-himpunannya.

a. Fungsi Keangotaan Sudut Bola

Pada Variabel Sudut Bola ada 3 macam himpunan fuzzy yaitu: KECIL, AGAK KECIL, SEDANG, AGAK BESAR, BESAR. Himpunan KECIL memiliki pendekatan fungsi keangotaan linier turun bahu kiri, Himpunan AGAK KECIL, AGAK BESAR menggunakan pendekatan fungsi keangotaan segitiga, untuk Himpunan SEDANG menggunakan fungsi keangotaan trapesium. untuk himpunan BESAR menggunakan pendekatan fungsi keangotaan linier naik bahu kanan.

u Sudut Bola



Gambar 3.6.1.1 Fungsi Keangotaan Sudut Bola

$$\mu_{sudut\ bola}KECIL(x) = \begin{cases} 1, & x < 70 \\ \dfrac{x-70}{80-70}, & 70 \le x \le 80 \\ 0, & x > 80 \end{cases}$$

$$\mu_{sudut\ bola}AGAK\ KECIL(x) = \begin{cases} 0, & x < 70\ atau\ x > 80 \\ \dfrac{80-x}{80-70}, & 70 \le x \le 80 \\ \dfrac{x-80}{89-80}, & 80 \le x \le 89 \end{cases}$$

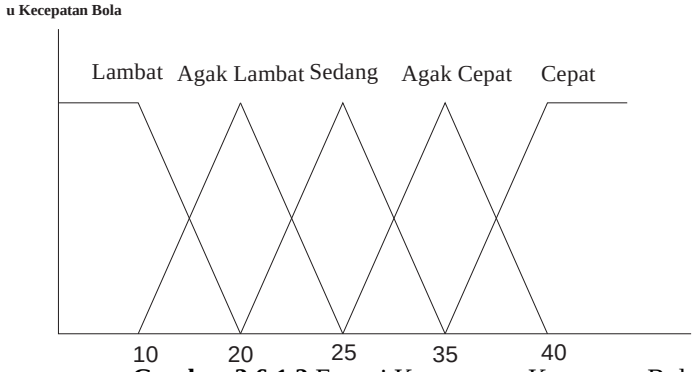
$$\mu_{sudut\ bola}SEDANG(x) = \begin{cases} 0, & x < 80\ atau\ x > 100 \\ \dfrac{89-x}{89-80}, & 80 \le x \le 89 \\ 1, & 89 \le x \le 91 \\ \dfrac{x-91}{100-91}, & 91 \le x \le 100 \end{cases}$$

$$\mu_{sudut\ bola}AGAK\ BESAR(x) = \begin{cases} 0, & x < 91\ atau\ x > 110 \\ \dfrac{100-x}{100-91}, & 91 \le x \le 100 \\ \dfrac{x-100}{110-100}, & 100 \le x \le 110 \end{cases}$$

$$\mu_{sudut\ bola}BESAR(x) = \begin{cases} 0, & x < 100 \\ \dfrac{110-x}{110-100}, & 100 \le x \le 110 \\ 1, & x > 110 \end{cases}$$

b. Fungsi Keangotaan Kecepatan Bola

Pada Variabel Sudut Bola ada 3 macam himpunan fuzzy yaitu: LAMBAT, AGAK LAMBAT, SEDANG, AGAK CEPAT, CEPAT. Himpunan LAMBAT memiliki pendekatan fungsi keangotaan linier turun bahu kiri, Himpunan AGAK LAMBAT, AGAK CEPAT menggunakan pendekatan fungsi keangotaan segitiga, untuk himpunan CEPAT menggunakan pendekatan fungsi keangotaan linier naik bahu kanan.



Gambar 3.6.1.2 Fungsi Keangotaan Kecepatan Bola

$$\mu_{kecepatan\ bola}LAMBAT(x) = \begin{cases} 1, & x < 10 \\ \dfrac{x-10}{20-10}, & 10 \le x \le 20 \\ 0, & x > 20 \end{cases}$$

$$\mu_{kecepatan\ bola}AGAK\ LAMBAT(x) = \begin{cases} 0, & x < 10\ atau\ x > 25 \\ \dfrac{20-x}{20-10}, & 10 \le x \le 20 \\ \dfrac{x-20}{25-20}, & 20 \le x \le 25 \end{cases}$$

$$\mu_{kecepatan\ bola}SEDANG(x) = \begin{cases} 0, & x < 20\ atau\ x > 35 \\ \dfrac{25-x}{25-20}, & 20 \le x \le 25 \\ \dfrac{x-25}{35-25}, & 25 \le x \le 35 \end{cases}$$

$$\mu_{kecepatan\ bola}AGAK\ CEPAT(x) = \begin{cases} 0, & x < 25\ atau\ x > 40 \\ \dfrac{35-x}{35-25}, & 25 \le x \le 35 \\ \dfrac{x-35}{40-35}, & 35 \le x \le 40 \end{cases}$$

$$\mu_{kecepatan\ bola}CEPAT(x) = \begin{cases} 0, & x < 35 \\ \dfrac{40-x}{40-35}, & 35 \le x \le 40 \\ 1, & x > 40 \end{cases}$$

3.6.2 Perancangan Rule

Berisi tentang aturan-aturan yang berlaku untuk semua kejadian (kombinasi). Proses ini berfungsi untuk mencari suatu nilai fuzzy output dari fuzzy input.

Prosesnya adalah sebagai berikut : suatu nilai fuzzy input yang berasal dari proses fuzzyfikasi kemudian dimasukkan kedalam sebuah rule yang telah dibuat untuk dijadikan sebuah fuzzy output. Berikut ini adalah Rule yang akan digunakan untuk perhitungan fuzzy:

Sudut\\\v	Lambat	Agak Lambat	Sedang	Agak Cepat	Cepat
Kecil	Kanan Sedang	Kanan Agak Cepat	Kanan Cepat	Kanan Cepat	Kanan Cepat
Agak Kecil	Kanan Lambat	Kanan Sedang	Kanan Agak Cepat	Kanan Cepat	Kanan Cepat
Sedang	Diam	Diam	Diam	Diam	Diam
Agak Besar	Kiri Lambat	Kiri Sedang	Kiri Agak Cepat	Kiri Cepat	Kiri Cepat
Besar	Kiri Sedang	Kiri Agak Cepat	Kiri Cepat	Kiri Cepat	Kiri Cepat

Tabel 3.6.2.1 Rule Fuzzy

KONDISI_GERAK_KANAN_CEPAT	40
KONDISI_GERAK_KANAN_AGAK_CEPAT	30
KONDISI_GERAK_KANAN_SEDANG	25
KONDISI_GERAK_KANAN_LAMBAT	10
KONDISI_GERAK_DIAM	0
KONDISI_GERAK_KIRI_LAMBAT	-10
KONDISI_GERAK_KIRI_SEDANG	-25
KONDISI_GERAK_KIRI_AGAK_CEPAT	-30
KONDISI_GERAK_KIRI_CEPAT	-40

Tabel 3.6.2.2 *Output Fuzzy*

Disini akan dicontohkan input yang masuk dan menghitung *output fuzzy*-nya. Dimisalkan sudut bola adalah 45 dan kecepatan bola adalah 30. Langkah pertama yang dilakukan kita harus memplot dulu inputnya di masing-masing variable fungsi keanggotaan.

Sudut bola = 45 maka termasuk dalam kategori kecil dan tidak masuk dalam fungsi keanggotaan lain.

$$\mu_{sudut\ bola} KECIL(45) = \begin{cases} 1, & 45 < 70 \\ \dfrac{x-70}{80-70}, & 70 \le x \le 80 \\ 0, & x > 80 \end{cases}$$

$$\mu_{sudut\ bola} KECIL(45) = 1$$

Kecepatan Bola = 30 maka termasuk dalam kategori sedang-agak cepat, maka:

$$\mu_{kecepatan\ bola}SEDANG(30) = \begin{cases} 0, & x < 20\ atau\ x > 35 \\ \dfrac{25-x}{25-20}, & 20 \le x \le 25 \\ \dfrac{x-25}{35-25}, & 25 \le x \le 35 \end{cases}$$

$$\mu_{kecepatan\ bola}SEDANG(30) = 0{,}5$$

$$\mu_{kecepatan\ bola}AGAK\ CEPAT(30) = \begin{cases} 0, & x < 25\ atau\ x > 40 \\ \dfrac{35-x}{35-25}, & 25 \le x \le 35 \\ \dfrac{x-35}{40-35}, & 35 \le x \le 40 \end{cases}$$

$$\mu_{kecepatan\ bola}AGAK\ CEPAT(30) = 0{,}5$$

Lalu mencari nilai minimum dari setiap himpunan yang selanjutnya akan dimasukkan dalam rule *Fuzzy*.

Sudut\\\v	Lambat = 0	Agak Lambat=0	Sedang = 0,5	Agak Cepat = 0,5	Cepat = 0
Kecil = 1	0	0	0,5	0,5	0
Agak Kecil = 0	0	0	0	0	0
Sedang = 0	0	0	0	0	0
Agak Besar = 0	0	0	0	0	0
Besar = 0	0	0	0	0	0

Lalu dalam proses *defuzzifikasi* derajat fungsi keanggotaan dikalikan dengan nilai output yang dikehendaki. Semua hasil perkalian tersebut dijumlahkan dan dibagi dengan jumlah total nilai minimum yang sudah didapat pada rule diatas, metode ini disebut rata-rata terbobot.

$$z = \frac{\sum \min(\mu_{sudut\ bola}, \mu_{kecepatan\ bola}) * OutputFuzzyRule}{\sum \min(\mu_{sudut\ bola}, \mu_{kecepatan\ bola})}$$

$$z = \frac{0{,}5 * 40 + 0{,}5 * 40}{0{,}5 + 0{,}5} = 40$$

Maka output keluaran fuzzyadalah 40 yang akan jadi kecepatan gerak terhadap sumbu x robot.

Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN

Pada bab ini berisi tentang pengujian dari sistem yang telah dibuat. Pengujian ini bertujuan untuk mengetahui tingkat ketercapaian tujuan dari sistem yang dirancang.

4.1 Uji Jarak Bola Terhadap Robot

Sebelum mendapatkan kecepatan dan sudut bola kita harus mencari posisi bola terhadap robot. Lalu setelah menemukan jarak tersebut kita dapat menemukan posisi bola pada lapangan. Berikut data hasil pengukuran tersebut. Data tersebut akan masuk ke regresi polinomial agar data pixel dapat sama dengan jarak dalam meter.

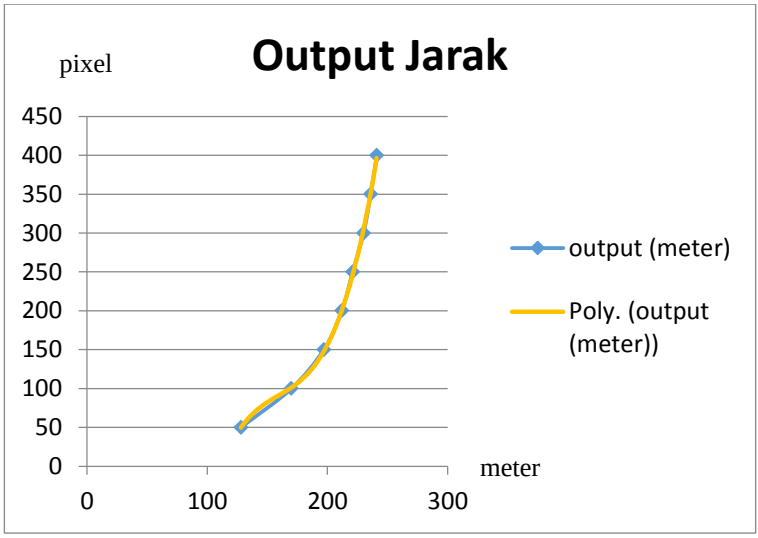
Input (pixel)	Output (meter)	Perhitungan Output	Error
128	50	49,79548876	0,2045112389
170	100	101,3221991	1,322199056
197	150	147,9076201	2,092379933
212	200	200,8446555	0,8446554676
221	250	246,5678476	3,432152372
230	300	305,0351495	5,035149497
236	350	352,0119497	2,01194968
241	400	396,5150898	3,484910176
Rata-rata Error			2,303488428

Tabel 4.1 Tabel Jarak Bola Terhadap Robot

Didapatkan persamaan regresi polonomial orde-2 dari data input ke data output sebagai berikut.

$$y = 0{,}0004x^3 - 0{,}2033x^2 + 33{,}292x - 1772{,}9$$

Data dari pengujian pendeteksian jarak bola direpresentasikan pada grafik berikut ini:



Gambar 4.1 Uji Pendeteksian Jarak Bola

Dari percobaan ini, sistem mampu 50 cm sampai dengan 400 cm. Dari grafik diatas antara input dan output hasil perhitungan hampir mendekati.

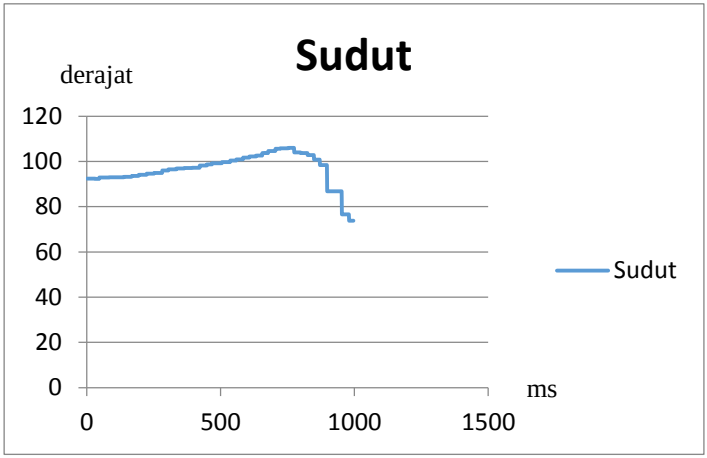
4.2 Pengujian Jalan Algoritma

Pada pengujian jalan menggunakan algoritma *Fuzzy* ada 3 pecobaan tendangan yang dilakukan. Tiap tendangan ditendang dengan arah yang berbeda. Tendangan pertama ditendang melalui tengah lapangan ke tengah gawang, tendangan kedua ditendang dari tengah lapangan ke kanan gawang. Tendangan ke 3 ditendang dari tengah lapangan menuju kiri gawang. Berikut hasil plot data sudut bola, kecepatan bola dan output *fuzzy*-nya.

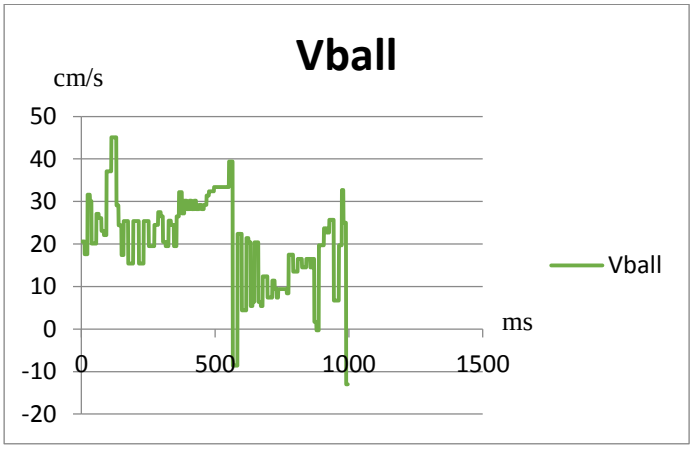
4.2.1 Data Hasil Tendangan Pertama (Dari tengah lapangan menuju ke tengah gawang)

Pada percobaan tendangan pertama, tendangan dilakukan dari tengah lapangan ke arah tengah gawang. Pada grafik sudut

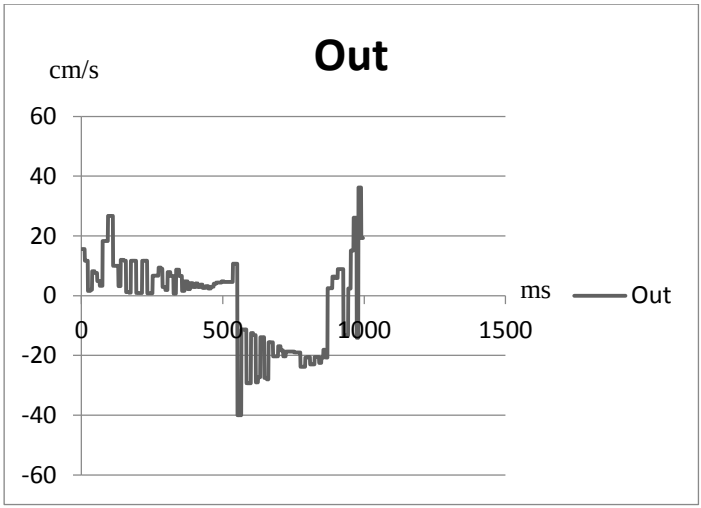
pada gambar di bawah dapat kita lihat pada waktu 0 sampai 500 ms sudut menunjukkan pergerakan bola sekitar kurang lebih 90 derajat dari robot yang berarti mengarah tepat kearah robot. Lalu pada waktu 500 hingga 800 ms, sudut mengarah sedikit ke arah kanan gawang. Lalu robot menuju ke arah kanan gawang hingga bola tertahan oleh robot.



Gambar 4.2.1.1 Plot Sudut Bola terhadap Robot



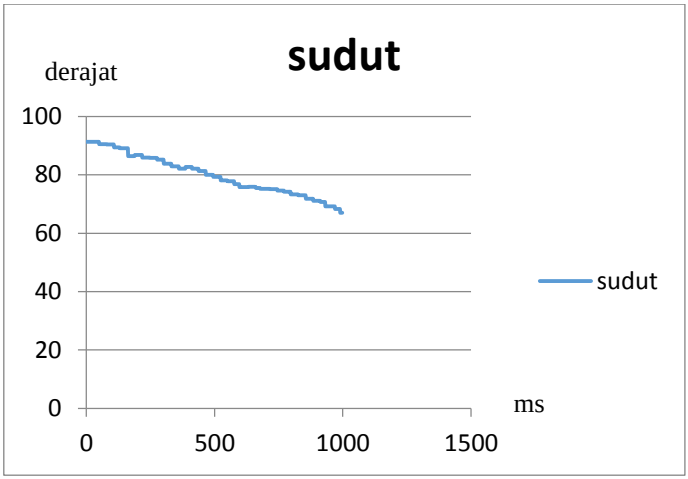
Gambar 4.2.1.2 Plot Kecepatan Bola



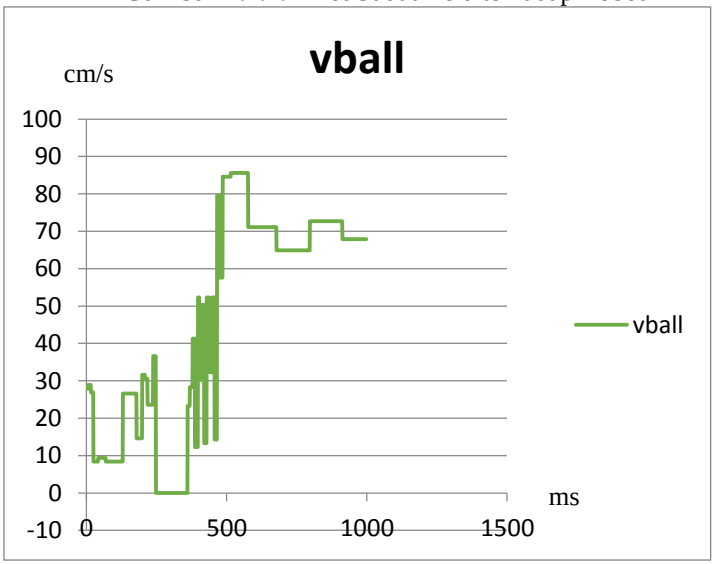
Gambar 4.2.1.3 Plot Output Fuzzy

4.2.2 Data Hasil Tendangan Kedua (Dari tengah lapangan menuju ke kanan gawang)

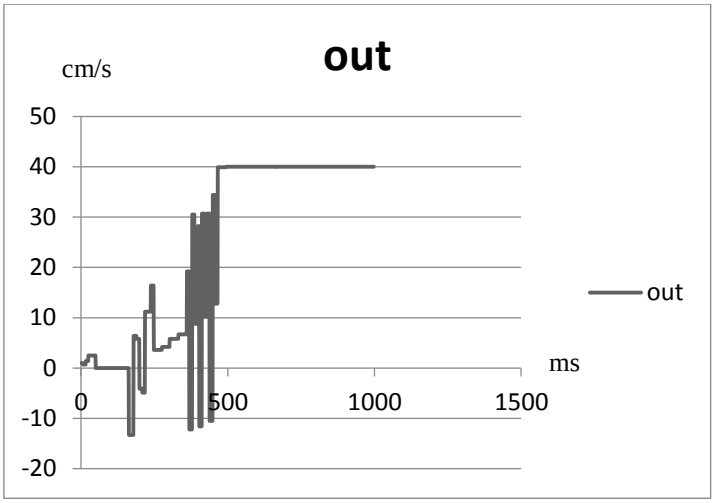
Pada percobaan tendangan kedua, tendangan dilakukan dari tengah lapangan ke arah kanan gawang. Pada grafik sudut pada gambar di bawah dapat kita lihat pada waktu 0 sampai 1000 ms sudut menunjukkan pergerakan bola dari 90 derajat ke sekitar 70 derajat terhadap robot yang berarti mengarah ke kanan gawang. Lalu robot menuju ke arah kanan gawang hingga bola tertahan oleh robot.



Gambar 4.2.2.1 Plot Sudut Bola terhadap Robot



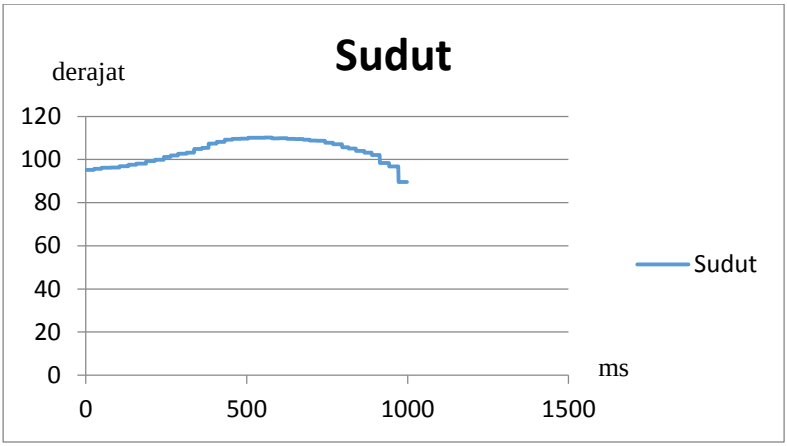
Gambar 4.2.2.2 Plot Kecepatan Bola



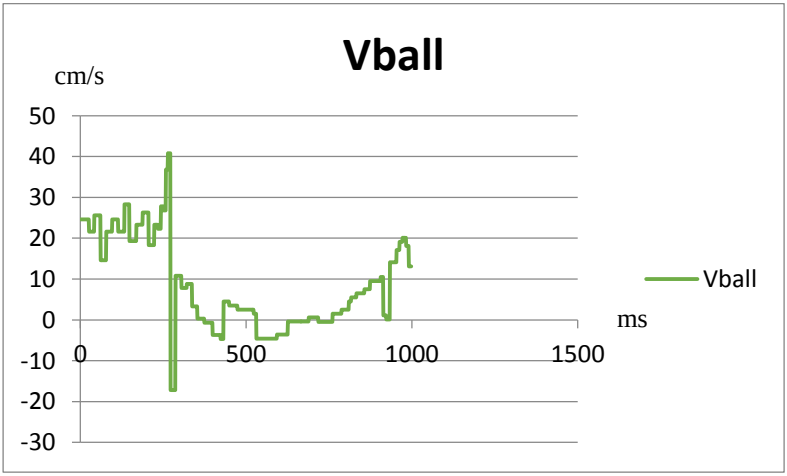
Gambar 4.2.2.3 Plot Output Fuzzy

4.2.3 Data Hasil Tendangan Ketiga (Dari tengah lapangan menuju ke kiri gawang)

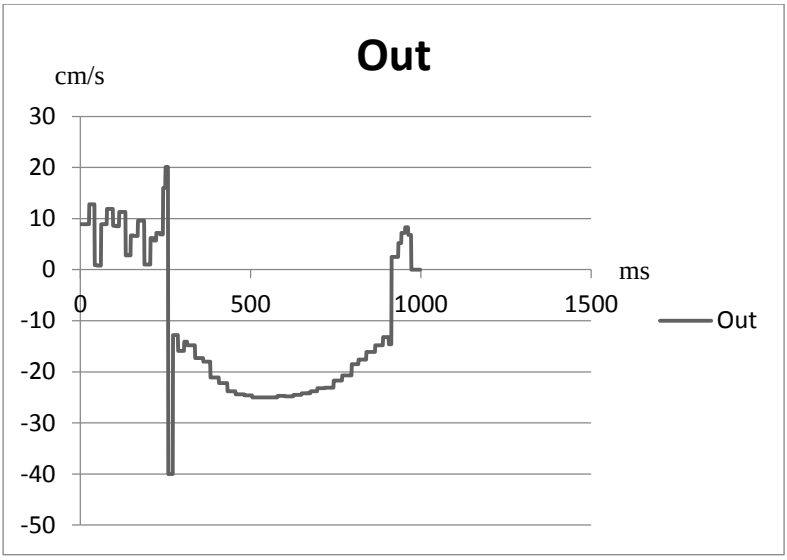
Pada percobaan tendangan ketiga, tendangan dilakukan dari tengah lapangan ke arah kiri gawang. Pada grafik sudut pada gambar di bawah dapat kita lihat pada waktu 0 sampai 700 ms sudut menunjukkan pergerakan bola dari 90 derajat ke sekitar 110 derajat terhadap robot yang berarti mengarah ke kiri gawang. Lalu robot menuju ke arah kiri gawang hingga bola tertahan oleh robot.



Gambar 4.2.3.1 Plot Sudut Bola terhadap Robot



Gambar 4.2.3.2 Plot Kecepatan Bola



Gambar 4.2.3.3 Plot Output Fuzzy

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang dapat diambil oleh penulis dari serangkaian pengujian yang telah dilakukan adalah sebagai berikut:

1. Metode penghadangan bola dengan metode Fuzzy Logic pada penjaga gawang sepak bola beroda dapat menyelamatkan tendangan kearah gawang dengan persentase 76%.
2. Pendeteksian bola menggunakan kamera omni direksional dapat mendeteksi bola pada jarak 50cm - 400cm. metode mengkonversi jarak dalam pixel ke meter menggunakan metode regresi polinomial orde-3 dapat mendekati jarak yang diinginkan. Kesalahan yang terjadi pada pembacaan memiliki rata-rata sebesar 2,3%.

5.2 Saran

Beberapa saran yang dapat diberikan penulis untuk pengembangan tugas akhir ini adalah sebagai berikut:

1. Untuk pendeteksian kecepatan menggunakan metode yang lebih baik dan kamera dengan kecepatan penangkapan gambar yang lebih tinggi agar data kecepatan bola tidak terjadi *glitch* atau loncatan error data yang tinggi.
2. Input metode Fuzzy Logic yang digunakan menggunakan input lain yang lebih mudah dan lebih cepat didapatkan, agar waktu pemrosesan bisa lebih cepat.
3. Penggunaan data sudut bola terhadap robot juga dirasa kurang cocok untuk keadaan tertentu, seperti posisi bola saat ditendang dari kiri lapangan. Robot akan menuju ke posisi bola di kiri lapangan dahulu dalam mengejar bola, setelah bola bergerak lebih cepat robot akan mengejar kearah kiri dahulu hingga melewati sudut tengah robot, lalu akan bermanuver ke kanan sesuai arah dan kecepatan bola saat akan masuk ke gawang. Metode ini dirasa kurang efektif. Kedepannya seharusnya dapat digunakan metode perpotongan titik bola terhadap sumbu x gawang/garis gawang secara virtual dengan prediksi trayektori, agar robot tidak harus bergerak seperti metode yang digunakan sekarang.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Anang Kholili, Hamdan. "Sistem Informasi Spare Part Mobil Dengan Fasilitas Estimasi Stok Menggunakan Fuzzy Tsukamoto". PENS-ITS, Surabaya: 2012.

[2] Kusumadewi, Sri. "Artificial Intelligence (Teknik dan Aplikasinya)". Graha Ilmu, Yogyakarta: 2003.

[3] Matlab. " *What's Fuzzy Logic*". 12 Mei 2018. https://www.mathworks.com/help/fuzzy/what-is-fuzzy-logic.html [diakses 12 Mei 2018]

[4] OpenCV, "OpenCV," OpenCV, 2018. [Online]. Available: http://opencv.org/. [Diakses 11 Mei 2018].

[5] openFramework, "openFramework," 14 Mei 2018. [Online]. Available: http://openframeworks.cc/. [Diakses 14 5 2018].

[6] T. InvenSense, "MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices," TDK InvenSense, 2018. [Online]. Available: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/. [Diakses 13 Mei 2018].

[7] STMicroelectronics,"STM32F4DISCOVERY," STMicroelectronics,2017.[Online].Available: http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf

[8] Corporation, "INTEL® NUC KIT NUC6I7KYK," Intel Corporation, [Online]. Available: https://www.intel.com/content/www/us/en/products/boards-kits/nuc/kits/nuc6i7kyk.html.

[9] Arduino, "Arduino NANO Pinout Diagram," Arduino, 2017. [Online].Available: https://forum.arduino.cc/index.php?topic=147582.0.

[10] Hermawanto, Denny. "Tutorial Pemrograman Fuzzy Logic". Bandung : 2008.

[11] Scaramuzza, D. (2008). Omnidirectional vision: from calibration to robot motion estimation, PhD thesis n. 17635. PhD thesis, ETH Zurich

[12] PlanetaryGearMotor-PGM455.2:1, "PlanetaryGearMotor", 2018, Available: http://www.lynxmotion.com/p-1073-planetary-gear-motor-pgm45-521.aspx

Halaman ini sengaja dikosongkan

LAMPIRAN

Program Visual studio

Fuzzy.h

```
#pragma once
#include<iostream>
#include "ofThread.h"

#define KONDISI_KECEPATAN_BOLA_LAMBAT            10
#define KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT  20
#define KONDISI_KECEPATAN_BOLA_SEDANG            25
#define KONDISI_KECEPATAN_BOLA_AGAK_CEPAT        35
#define KONDISI_KECEPATAN_BOLA_CEPAT             40

#define KONDISI_SUDUT_BOLA_KECIL      70
#define KONDISI_SUDUT_BOLA_AGAK_KECIL 80
#define KONDISI_SUDUT_BOLA_SEDANG1         89
#define KONDISI_SUDUT_BOLA_SEDANG2         91
#define KONDISI_SUDUT_BOLA_AGAK_BESAR 100
#define KONDISI_SUDUT_BOLA_BESAR   110


#define KONDISI_GERAK_KANAN_CEPAT 40
#define KONDISI_GERAK_KANAN_AGAK_CEPAT 30
#define KONDISI_GERAK_KANAN_SEDANG 25
#define KONDISI_GERAK_KANAN_LAMBAT 10
#define KONDISI_GERAK_DIAM 0
#define KONDISI_GERAK_KIRI_LAMBAT -10
#define KONDISI_GERAK_KIRI_SEDANG -25
#define KONDISI_GERAK_KIRI_AGAK_CEPAT -30
#define KONDISI_GERAK_KIRI_CEPAT -40

using namespace std;

class Fuzzy : public ofThread
{
private:
        float derajat_ball_tetha[7];
        float derajat_kecepatan_bola[7];
        float temp;
        float aturan00, aturan01, aturan02, aturan03, aturan04;
        float aturan10, aturan11, aturan12, aturan13, aturan14;
        float aturan20, aturan21, aturan22, aturan23, aturan24;
        float aturan30, aturan31, aturan32, aturan33, aturan34;
        float aturan40, aturan41, aturan42, aturan43, aturan44;
```

```
        float rule[5][5];
public:
        Fuzzy();
        ~Fuzzy();
        float z;
        float kecepatan_bola;
        float posisi_y, posisi_x;
        float ball_tetha;
        float ball_x, ball_y;
        void MembershipSudutBola();
        void MembershipKecepatanBola();
        void Tsukamoto();
        void Defuzzy();
        void threadedFunction();
};
```

Fuzzy.cpp

```
#include "Fuzzy.h"
#include <math.h>

Fuzzy::Fuzzy()
{

}

Fuzzy::~Fuzzy()
{

}

void Fuzzy::threadedFunction()
{
	while (isThreadRunning())
	{
		//cout << ball_tetha << "\t" << kecepatan_bola <<
endl;
		Defuzzy();
	}
}
void Fuzzy::MembershipKecepatanBola()
{
	// untuk kecepatan LAMBAT
	if (kecepatan_bola <= KONDISI_KECEPATAN_BOLA_LAMBAT)
	{
		derajat_kecepatan_bola[0] = 1;
	}
	else if (kecepatan_bola > KONDISI_KECEPATAN_BOLA_LAMBAT &&
kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT)
	{
		derajat_kecepatan_bola[0] =
(KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT - kecepatan_bola) /
(KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT -
KONDISI_KECEPATAN_BOLA_LAMBAT);
	}
	else
	{
		derajat_kecepatan_bola[0] = 0;
	}

	// untuk kecepatan agak LAMBAT
	if (kecepatan_bola <= KONDISI_KECEPATAN_BOLA_LAMBAT)
	{
		derajat_kecepatan_bola[1] = 0;
```

```
}
else if (kecepatan_bola > KONDISI_KECEPATAN_BOLA_LAMBAT &&
kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT)
{
derajat_kecepatan_bola[1] = (kecepatan_bola –
KONDISI_KECEPATAN_BOLA_LAMBAT) /
(KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT –
KONDISI_KECEPATAN_BOLA_LAMBAT);
}
else if (kecepatan_bola >
KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT && kecepatan_bola <=
KONDISI_KECEPATAN_BOLA_SEDANG)
{
derajat_kecepatan_bola[1] = (KONDISI_KECEPATAN_BOLA_SEDANG
- kecepatan_bola) / (KONDISI_KECEPATAN_BOLA_SEDANG –
KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT);
}
else
{
derajat_kecepatan_bola[1] = 0;
}

// untuk kecepatan sedang
if (kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT)
{
        derajat_kecepatan_bola[2] = 0;
}
else if (kecepatan_bola >
KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT && kecepatan_bola <=
KONDISI_KECEPATAN_BOLA_SEDANG)
{
derajat_kecepatan_bola[2] = (kecepatan_bola –
KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT) /
(KONDISI_KECEPATAN_BOLA_SEDANG –
KONDISI_KECEPATAN_BOLA_AGAK_LAMBAT);
}
else if (kecepatan_bola > KONDISI_KECEPATAN_BOLA_SEDANG &&
kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_CEPAT)
{
        derajat_kecepatan_bola[2] =
(KONDISI_KECEPATAN_BOLA_SEDANG - kecepatan_bola) /
(KONDISI_KECEPATAN_BOLA_AGAK_CEPAT –
KONDISI_KECEPATAN_BOLA_SEDANG);
}
else
{
derajat_kecepatan_bola[2] = 0;
}
```

```
        // untuk kecepatan agak CEPAT
        if (kecepatan_bola <= KONDISI_KECEPATAN_BOLA_SEDANG)
        {
                derajat_kecepatan_bola[3] = 0;
        }
        else if (kecepatan_bola > KONDISI_KECEPATAN_BOLA_SEDANG &&
kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_CEPAT)
        {
                derajat_kecepatan_bola[3] = (kecepatan_bola -
KONDISI_KECEPATAN_BOLA_SEDANG) /
(KONDISI_KECEPATAN_BOLA_AGAK_CEPAT -
KONDISI_KECEPATAN_BOLA_SEDANG);
        }
        else if (kecepatan_bola >
KONDISI_KECEPATAN_BOLA_AGAK_CEPAT && kecepatan_bola <=
KONDISI_KECEPATAN_BOLA_CEPAT)
        {
                derajat_kecepatan_bola[3] =
(KONDISI_KECEPATAN_BOLA_CEPAT - kecepatan_bola) /
(KONDISI_KECEPATAN_BOLA_CEPAT -
KONDISI_KECEPATAN_BOLA_AGAK_CEPAT);
        }
        else
        {
                derajat_kecepatan_bola[3] = 0;
        }

        // untuk kecepatan CEPAT
        if (kecepatan_bola <= KONDISI_KECEPATAN_BOLA_AGAK_CEPAT)
        {
                derajat_kecepatan_bola[4] = 0;
        }
        else if (kecepatan_bola >
KONDISI_KECEPATAN_BOLA_AGAK_CEPAT && kecepatan_bola <=
KONDISI_KECEPATAN_BOLA_CEPAT)
        {
                derajat_kecepatan_bola[4] = (kecepatan_bola -
KONDISI_KECEPATAN_BOLA_AGAK_CEPAT) / (KONDISI_KECEPATAN_BOLA_CEPAT
- KONDISI_KECEPATAN_BOLA_AGAK_CEPAT);
        }
        else if (kecepatan_bola > KONDISI_KECEPATAN_BOLA_CEPAT)
        {
                derajat_kecepatan_bola[4] = 1;
        }
}

void Fuzzy::MembershipSudutBola()
```

```
{
        // untuk sudut KECIL
        if (ball_tetha <= KONDISI_SUDUT_BOLA_KECIL)
        {
                derajat_ball_tetha[0] = 1;
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_KECIL &&
ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_KECIL)
        {
                derajat_ball_tetha[0] =
(KONDISI_SUDUT_BOLA_AGAK_KECIL - ball_tetha) /
(KONDISI_SUDUT_BOLA_AGAK_KECIL - KONDISI_SUDUT_BOLA_KECIL);
        }
        else
        {
                derajat_ball_tetha[0] = 0;
        }

        // untuk sudut agak KECIL
        if (ball_tetha <= KONDISI_SUDUT_BOLA_KECIL)
        {
                derajat_ball_tetha[1] = 0;
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_KECIL &&
ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_KECIL)
        {
                derajat_ball_tetha[1] = (ball_tetha -
KONDISI_SUDUT_BOLA_KECIL) / (KONDISI_SUDUT_BOLA_AGAK_KECIL -
KONDISI_SUDUT_BOLA_KECIL);
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_AGAK_KECIL &&
ball_tetha <= KONDISI_SUDUT_BOLA_SEDANG1)
        {
                derajat_ball_tetha[1] =
(KONDISI_SUDUT_BOLA_SEDANG1 - ball_tetha) /
(KONDISI_SUDUT_BOLA_SEDANG1 - KONDISI_SUDUT_BOLA_AGAK_KECIL);
        }
        else
        {
                derajat_ball_tetha[1] = 0;
        }

        // untuk sudut sedang(TRAPEZOID)
        if (ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_KECIL)
        {
                derajat_ball_tetha[2] = 0;
        }
```

```
        else if (ball_tetha > KONDISI_SUDUT_BOLA_AGAK_KECIL &&
ball_tetha <= KONDISI_SUDUT_BOLA_SEDANG1)
        {
                derajat_ball_tetha[2] = (ball_tetha -
KONDISI_SUDUT_BOLA_AGAK_KECIL) / (KONDISI_SUDUT_BOLA_SEDANG1 -
KONDISI_SUDUT_BOLA_AGAK_KECIL);
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_SEDANG1 &&
ball_tetha <= KONDISI_SUDUT_BOLA_SEDANG2)
        {
                derajat_ball_tetha[2] = 1;
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_SEDANG2 &&
ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_BESAR)
        {
                derajat_ball_tetha[2] =
(KONDISI_SUDUT_BOLA_SEDANG2 - ball_tetha) /
(KONDISI_SUDUT_BOLA_AGAK_BESAR - KONDISI_SUDUT_BOLA_SEDANG2);
        }
        else
        {
                derajat_ball_tetha[2] = 0;
        }

        // untuk sudut agak BESAR
        if (ball_tetha <= KONDISI_SUDUT_BOLA_SEDANG2)
        {
                derajat_ball_tetha[3] = 0;
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_SEDANG2 &&
ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_BESAR)
        {
                derajat_ball_tetha[3] = (ball_tetha -
KONDISI_SUDUT_BOLA_SEDANG2) / (KONDISI_SUDUT_BOLA_AGAK_BESAR -
KONDISI_SUDUT_BOLA_SEDANG2);
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_AGAK_BESAR &&
ball_tetha <= KONDISI_SUDUT_BOLA_BESAR)
        {
                derajat_ball_tetha[3] = (KONDISI_SUDUT_BOLA_BESAR
- ball_tetha) / (KONDISI_SUDUT_BOLA_BESAR -
KONDISI_SUDUT_BOLA_AGAK_BESAR);
        }
        else
        {
                derajat_ball_tetha[3] = 0;
        }
```

```
        // untuk sudut BESAR
        if (ball_tetha <= KONDISI_SUDUT_BOLA_AGAK_BESAR)
        {
                derajat_ball_tetha[4] = 0;
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_AGAK_BESAR &&
ball_tetha <= KONDISI_SUDUT_BOLA_BESAR)
        {
                derajat_ball_tetha[4] = (ball_tetha -
KONDISI_SUDUT_BOLA_AGAK_BESAR) / (KONDISI_SUDUT_BOLA_BESAR -
KONDISI_SUDUT_BOLA_AGAK_BESAR);
        }
        else if (ball_tetha > KONDISI_SUDUT_BOLA_BESAR)
        {
                derajat_ball_tetha[4] = 1;
        }
}

void Fuzzy::Tsukamoto()
{
        int i, j;
        MembershipKecepatanBola();
        MembershipSudutBola();
        for (i = 0; i <= 4; i++)
        {
                for (j = 0; j <= 4; j++)
                {
                        //FUNGSI AND (mencari yang terkecil)
                        if (derajat_ball_tetha[i] >
derajat_kecepatan_bola[j])
                        {
                                temp = derajat_kecepatan_bola[j];
                        }
                        else
                        {
                                temp = derajat_ball_tetha[i];
                        }
                        rule[i][j] = temp;
                }
        }
aturan00 = rule[0][0]; // (kecil,lambat = kanan sedang)
aturan01 = rule[0][1]; // (kecil,agak lambat = kanan agak cepat)
aturan02 = rule[0][2]; // (kecil,sedang = kanan cepat)
aturan03 = rule[0][3]; // (kecil,agak cepat = kanan cepat)
aturan04 = rule[0][4]; // (kecil,cepat = kanan cepat)

aturan10 = rule[1][0]; // (agak kecil,lambat  = kanan lambat)
aturan11 = rule[1][1]; // (agak kecil,agak lambat = kanan sedang)
```

```
aturan12 = rule[1][2]; // (agak kecil,sedang  = kanan agak cepat)
aturan13 = rule[1][3]; // (agak kecil,agak cepat  = kanan cepat)
aturan14 = rule[1][4]; // (agak kecil,cepat = kanan cepat)

aturan20 = rule[2][0]; // (sedang,lambat = stop)
aturan21 = rule[2][1]; // (sedang,agak lambat = stop)
aturan22 = rule[2][2]; // (sedang,sedang= stop)
aturan23 = rule[2][3]; // (sedang,agak cepat = stop)
aturan24 = rule[2][4]; // (sedang,cepat = stop)

aturan30 = rule[3][0]; // (agak besar,lambat = kiri lambat)
aturan31 = rule[3][1]; // (agak besar,agak lambat = kiri sedang)
aturan32 = rule[3][2]; // (agak besar,sedang=  kiri agak cepat)
aturan33 = rule[3][3]; // (agak besar,agak cepat =  kiri cepat)
aturan34 = rule[3][4]; // (agak besar,cepat = kiri cepat)

aturan40 = rule[4][0]; // (besar,lambat = kiri sedang)
aturan41 = rule[4][1]; // (besar,agak lambat = kiri agak cepat)
aturan42 = rule[4][2]; // (besar,sedang= kiri cepat)
aturan43 = rule[4][3]; // (besar,agak cepat = kiri cepat)
aturan44 = rule[4][4]; // (besar,cepat = kiri cepat)

}

void Fuzzy::Defuzzy()
{
        float defuz;
        Tsukamoto();
        z =     (aturan00 *  KONDISI_GERAK_KANAN_SEDANG) +
                (aturan01 *  KONDISI_GERAK_KANAN_AGAK_CEPAT) +
                (aturan02 *  KONDISI_GERAK_KANAN_CEPAT) +
                (aturan03 *  KONDISI_GERAK_KANAN_CEPAT) +
                (aturan04 *  KONDISI_GERAK_KANAN_CEPAT) +

                (aturan10 *  KONDISI_GERAK_KANAN_LAMBAT) +
                (aturan11 *  KONDISI_GERAK_KANAN_SEDANG) +
                (aturan12 *  KONDISI_GERAK_KANAN_AGAK_CEPAT) +
                (aturan13 *  KONDISI_GERAK_KANAN_CEPAT) +
                (aturan14 *  KONDISI_GERAK_KANAN_CEPAT) +

                (aturan20 *  KONDISI_GERAK_DIAM) +
                (aturan21 *  KONDISI_GERAK_DIAM) +
                (aturan22 *  KONDISI_GERAK_DIAM) +
                (aturan23 *  KONDISI_GERAK_DIAM) +
                (aturan24 *  KONDISI_GERAK_DIAM) +

                (aturan30 *  KONDISI_GERAK_KIRI_LAMBAT) +
                (aturan31 *  KONDISI_GERAK_KIRI_SEDANG) +
```

```
                (aturan32 *  KONDISI_GERAK_KIRI_AGAK_CEPAT) +
                (aturan33 *  KONDISI_GERAK_KIRI_CEPAT) +
                (aturan34 *  KONDISI_GERAK_KIRI_CEPAT) +

                (aturan40 *  KONDISI_GERAK_KIRI_SEDANG) +
                (aturan41 *  KONDISI_GERAK_KIRI_AGAK_CEPAT) +
                (aturan42 *  KONDISI_GERAK_KIRI_CEPAT) +
                (aturan43 *  KONDISI_GERAK_KIRI_CEPAT) +
                (aturan44 *  KONDISI_GERAK_KIRI_CEPAT);

        defuz = 0;
        int i, j;
        for (i = 0; i <= 4; i++)
        {
                for (j = 0; j <= 4; j++)
                {
                        defuz = defuz + rule[i][j];
                }
        }
        z = z / defuz;
        char buffer[100];
        sprintf(buffer, "%04.1f %04.1f %04.1f", ball_tetha,
kecepatan_bola, z);
        cout << buffer <<endl;
}
```

vision.h

```
#pragma once

#include "ofMain.h"
#include "ofxNetwork.h"
#include <opencv2\opencv.hpp>

using namespace cv;

class vision : public ofThread
{
public:
	vision();
	~vision();
	void setup_omnivision(int);
	void setup_frontvision(int);
	VideoCapture cam, cam_front;
	VideoWriter recorder;
	ofstream logger;

	//-----Informasi input
	int vx_avoid, vy_avoid;

	unsigned char status_obstacle_avoidance;

	float gyro_derajat;
	float gyro_radian;
	float posisi_x;
	float posisi_y;

	float posisi_x_global;
	float posisi_y_global;

	unsigned char rec_number;
	unsigned char rec_status;

	float cnn_activation_x[13];
	float cnn_activation_y[17];


	//-----Informasi output
	unsigned char status_bola;
	float sudut_bola;
	float bola_x_pada_frame;
	float bola_y_pada_frame;
	float bola_x_pada_lapangan;
	float bola_y_pada_lapangan;
```

```
        float bola_x0, bola_x1;
        float bola_y0, bola_y1;
        unsigned char arah_bola;
        float kecepatan_bola;
        float kecepatan_bola_x;
        float kecepatan_bola_y;

        int ball_x_frame_front;
        int ball_y_frame_front;
        float ball_x_pada_lapangan_front;
        float ball_y_pada_lapangan_front;
        int ball_d_front;

        unsigned char status_hindar_obstacle;
        unsigned char status_hindar_obstacle_ball;
        float   sudut_hindar_obstacle,   sudut_halangan_obstacle,
jarak_halangan_obstacle;
        float                          sudut_hindar_obstacle_ball,
sudut_halangan_obstacle_ball, jarak_halangan_obstacle_ball;

        float halangan_pada_frame[36];
        float halangan_pada_lapangan[36];

        char cnn_pixel[43200];

private:
        void threadedFunction();
        void proses_omnivision();
        void proses_frontvision();

        //-----Prototype
        void show_display();
        void show_display_front();

        void record_display();

        //---------------------------------
        void select_field_omnivision();
        void select_ball_omnivision();
        void select_field_front();
        void select_ball_front();
        void select_obstacle();
        void find_ball_front(Mat, Mat, Mat &_frame);
        //---------------------------------

        void calculate_obstacle();
        void calculate_obstacle_ball();
        //---------------------------------
```

```
        float pixel_to_cm(float _pixel);
        float pixel_to_cm_front(float _pixel);
        //--------------------------------
        void cnn_feedforward();
        void cnn_display();

        //-----Lapangan
        int hl_field, sl_field, vl_field;
        int hh_field, sh_field, vh_field;
        int opening_field_front, closing_field_front;
        Mat field_front;

        //-----Bola
        int hl_ball, sl_ball, vl_ball;
        int hh_ball, sh_ball, vh_ball;
        int opening_ball_front, closing_ball_front;
        Mat ball_front;
        float jarak_bola_ke_robot_frame_front;
        float jarak_bola_ke_robot_front;

        //-----Informasi Bola
        Point2f ball_center;
        float ball_radius;

        //-----Variabel penolong
        int offset_sudut = 0;    //Penting    untuk    keselamatan
penerbangan  ->  offset  sudut  jika  kamera  tidak  benar-benar
menghadap depan
        int offset_x = 0;               //Penting           untuk
keselamatan penerbangan -> offset kanan kiri untuk menengahkan
kamera
        int offset_y = 0;               //Penting           untuk
keselamatan penerbangan -> offset depan belakang untuk menegahkan
kamera

        int center_x;
        int center_y;
        int frame_width;
        int frame_height;

        int center_x_front;
        int center_y_front;
        int frame_width_front;
        int frame_height_front;

        //-----Pre-Process
        Rect roi;
        Mat source_omni, frame, frame_hsv, frame_gray;
```

```
        Mat source_front, source_front_hsv;

        int blur_front;

        //-----Result
        Mat field, field_threshold, raw_field_threshold;
        Mat ball, ball_threshold, raw_ball_threshold;
        Mat obstacle, obstacle_threshold;

        //-----FPS
        float fps;
        float fps_front;

        //int fps;
        uint64_t fps_1;
        uint64_t fps_0;
        uint64_t fps_front_1;
        uint64_t fps_front_0;

        //-----Display
        Mat display;
        Mat display_front;

        //-----Time
        uint64_t timenow;
        uint64_t timeprev;

};
```

vision.cpp

```
#include "vision.h"

vision::vision()
{
	ifstream ConfigFile;
	ConfigFile.open("C:\\IRIS\\config_vision.ini");
	if (ConfigFile.is_open())
	{
		ConfigFile >> hl_field;
		ConfigFile >> hh_field;
		ConfigFile >> sl_field;
		ConfigFile >> sh_field;
		ConfigFile >> vl_field;
		ConfigFile >> vh_field;
		ConfigFile >> hl_ball;
		ConfigFile >> hh_ball;
		ConfigFile >> sl_ball;
		ConfigFile >> sh_ball;
		ConfigFile >> vl_ball;
		ConfigFile >> vh_ball;
		ConfigFile >> opening_ball_front;
		ConfigFile >> closing_ball_front;
		ConfigFile >> opening_field_front;
		ConfigFile >> closing_field_front;
		ConfigFile.close();
	}
}

vision::~vision()
{
	ofstream ConfigFile;
	ConfigFile.open("C:\\IRIS\\config_vision.ini");
	if (ConfigFile.is_open())
	{
		ConfigFile << hl_field << endl;
		ConfigFile << hh_field << endl;
		ConfigFile << sl_field << endl;
		ConfigFile << sh_field << endl;
		ConfigFile << vl_field << endl;
		ConfigFile << vh_field << endl;
		ConfigFile << hl_ball << endl;
		ConfigFile << hh_ball << endl;
		ConfigFile << sl_ball << endl;
		ConfigFile << sh_ball << endl;
		ConfigFile << vl_ball << endl;
		ConfigFile << vh_ball << endl;
```

```
                ConfigFile << opening_ball_front << endl;
                ConfigFile << closing_ball_front << endl;
                ConfigFile << opening_field_front << endl;
                ConfigFile << closing_field_front << endl;
                ConfigFile.close();
        }
}

void vision::setup_omnivision(int index)
{
        cam.open(index);
        cam.read(source_omni);

        cam.set(CAP_PROP_ZOOM, 100);

        rotate(source_omni, source_omni, ROTATE_90_CLOCKWISE);
        flip(source_omni, source_omni, 1);

        if (offset_x >= 0) roi.x = offset_x * 2; else roi.x = 0;
        if (offset_y >= 0) roi.y = offset_y * 2; else roi.y = 0;
        roi.width = 480 - abs(offset_x * 2);
        roi.height = 640 - abs(offset_y * 2);

        frame = source_omni(roi);
        resize(frame, frame, Size(480, 640));

        center_x = frame.size().width / 2;
        center_y = frame.size().height / 2;
        frame_width = frame.size().width;
        frame_height = frame.size().height;

        namedWindow("Control", WINDOW_NORMAL);
        resizeWindow("Control",  300,  800);  moveWindow("Control",
        0, 0);
        createTrackbar("Lapangan LH", "Control", &hl_field, 180);
        createTrackbar("Lapangan HH", "Control", &hh_field, 180);
        createTrackbar("Lapangan LS", "Control", &sl_field, 255);
        createTrackbar("Lapangan HS", "Control", &sh_field, 255);
        createTrackbar("Lapangan LV", "Control", &vl_field, 255);
        createTrackbar("Lapangan HV", "Control", &vh_field, 255);
        createTrackbar("Bola LH", "Control", &hl_ball, 180);
        createTrackbar("Bola HH", "Control", &hh_ball, 180);
        createTrackbar("Bola LS", "Control", &sl_ball, 255);
        createTrackbar("Bola HS", "Control", &sh_ball, 255);
        createTrackbar("Bola LV", "Control", &vl_ball, 255);
        createTrackbar("Bola HV", "Control", &vh_ball, 255);
        createTrackbar("Op Ball", "Control", &opening_ball_front,
        15);
```

```
        createTrackbar("Cl Ball", "Control", &closing_ball_front,
        15);
        createTrackbar("Op Field", "Control",&opening_field_front,
        15);
        createTrackbar("Cl Field", "Control",&closing_field_front,
        15);

        namedWindow("Display", WINDOW_NORMAL);
        resizeWindow("Display", 240, 320);
        moveWindow("Display", 310, 0);
        namedWindow("Lapangan", WINDOW_NORMAL);
        resizeWindow("Lapangan", 120, 160);
        moveWindow("Lapangan", 560, 0);
        namedWindow("Bola", WINDOW_NORMAL);
        resizeWindow("Bola", 120, 160);
        moveWindow("Bola", 700, 0);
}

void vision::threadedFunction()
{
        while (isThreadRunning())
        {
                proses_omnivision();
                //proses_frontvision();

                select_field_omnivision();
                select_ball_omnivision();

                show_display();

                waitKey(1);
        }
}

void vision::proses_omnivision()
{
        cam.read(source_omni);
                        //Membaca gambar dari kamera omni
        rotate(source_omni,   source_omni,   ROTATE_90_CLOCKWISE);
        //Memutar gambar 90 derajat
        flip(source_omni, source_omni, 1);
        //Membalik gambar horizontal agar sesuai dengan robot
        frame = source_omni(roi);
        //Mengambil gambar yang sudah diberi offset
        resize(frame,  frame,  Size(frame_width,  frame_height));
        //Mengubah ukuran kembali menjadi 480x640 pixel
        Mat M = getRotationMatrix2D(Point(center_x, center_y),
```

```
        offset_sudut,  1.05);  //Membuat  matriks  rotasi  dengan
        perbesaran 5%
        warpAffine(frame, frame, M, Size(frame_width,
        frame_height));//Merotasi gambar untuk menyesuaikan
        pemasangan kamera
        circle(frame, Point(center_x, center_y), 80, Scalar(0, 0,
        0), -1);//Menutupi bagian yang tidak perlu (tengah)
        circle(frame, Point(center_x, center_y), 400, Scalar(0, 0,
        0), 90);//Menutupi bagian yang tidak perlu (tepi)
        //line(frame,  Point(center_x,  center_y),  Point(0,  550),
        Scalar(0, 0, 0), 180);  //Menutupi bagian yang tidak perlu
        (belakang)
        //line(frame, Point(center_x, center_y), Point(600, 700),
        Scalar(0, 0, 0), 180);  //Menutupi bagian yang tidak perlu
        (belakang)
        //line(frame, Point(0, 550), Point(480, 550), Scalar(0, 0,
        0), 300);  //Menutupi bagian yang tidak perlu (belakang)
        cvtColor(frame, frame_hsv, CV_BGR2HSV);
        //Mengubah gambar ke format HSV
        cvtColor(frame,          frame_gray,          CV_BGR2GRAY);
        //Mengubah gambar ke format GRAY

        display = frame.clone();
}

void vision::select_field_omnivision()
{

        if (hl_field > hh_field)
        {
        Mat temp_hsv; resize(frame_hsv, temp_hsv, Size(120, 160));
        Mat dstA; inRange(temp_hsv, Scalar(hl_field, sl_field,
        vl_field), Scalar(180, sh_field, vh_field), dstA);
        Mat dstB; inRange(temp_hsv, Scalar(0, sl_field, vl_field),
        Scalar(hh_field, sh_field, vh_field), dstB);
        bitwise_or(dstA, dstB, field_threshold);
        resize(field_threshold, field_threshold, Size(480, 640));
        threshold(field_threshold, field_threshold, 127, 255,
        THRESH_BINARY);
        }
        else
        {
        Mat temp_hsv; resize(frame_hsv, temp_hsv, Size(120, 160));
        inRange(temp_hsv, Scalar(hl_field, sl_field, vl_field),
        Scalar(hh_field, sh_field, vh_field), field_threshold);
        resize(field_threshold, field_threshold, Size(480, 640));
        threshold(field_threshold, field_threshold, 127, 255,
        THRESH_BINARY);
```

```
        }

        // ***** Menganggap lingkaran tengah sebagai lapangan

        circle(field_threshold, Point(center_x, center_y), 75,
        Scalar(255), -1);
        dilate(field_threshold,raw_field_threshold,
        getStructuringElement(MORPH_ELLIPSE, Size(11, 11)));
        imshow("Lapangan", raw_field_threshold);

        //-----Mencari Contour
        vector<Mat> contours;
        vector<Vec4i> hierarchy;
        findContours(raw_field_threshold, contours, hierarchy,
        CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);

        //-----Mencari Hull
        vector<Mat> hull(contours.size());
        for (int i = 0; i < contours.size(); i++)
                convexHull(Mat(contours[i]), hull[i]);

        //-----Mencari Hull Terbesar
        int largest_hull_index = 0;
        int largest_hull_area = 0;
        for (int i = 0; i < hull.size(); i++)
                if (contourArea(hull[i]) > largest_hull_area)
                {
                        largest_hull_area = contourArea(hull[i]);
                        largest_hull_index = i;
                }

        field_threshold = Scalar(0);
        drawContours(field_threshold, hull, largest_hull_index,
        Scalar(255), -1);

        // ***** Menggambar contour lapangan pada frame
        drawContours(display, hull, largest_hull_index,
        Scalar(255, 255, 0), 2);
}

void vision::select_ball_omnivision()
{

        if (hl_ball > hh_ball)
        {
        Mat temp_hsv; resize(frame_hsv, temp_hsv, Size(120, 160));
        Mat dstA; inRange(temp_hsv, Scalar(hl_ball, sl_ball,
        vl_ball), Scalar(180, sh_ball, vh_ball), dstA);
```

```
Mat dstB; inRange(temp_hsv, Scalar(0, sl_ball, vl_ball),
Scalar(hh_ball, sh_ball, vh_ball), dstB);
bitwise_or(dstA, dstB, ball_threshold);
resize(ball_threshold, ball_threshold, Size(480, 640));
threshold(ball_threshold,   ball_threshold,   127,   255,
THRESH_BINARY);
}
else
{
Mat temp_hsv; resize(frame_hsv, temp_hsv, Size(120, 160));
inRange(temp_hsv,   Scalar(hl_ball,   sl_ball,   vl_ball),
Scalar(hh_ball, sh_ball, vh_ball), ball_threshold);
resize(ball_threshold, ball_threshold, Size(480, 640));
threshold(ball_threshold, ball_threshold, 127, 255,
THRESH_BINARY);
}

dilate(ball_threshold, ball_threshold,
getStructuringElement(MORPH_ELLIPSE, Size(11, 11)));

// ***** Hanya melihat bola yang ada di dalam lapangan
bitwise_and(field_threshold, ball_threshold,
raw_ball_threshold);
imshow("Bola", raw_ball_threshold);

//-----Mencari Contour
vector<Mat> contours;
vector<Vec4i> hierarchy;
findContours(raw_ball_threshold, contours, hierarchy,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);

//-----Mencari Hull
vector<Mat> hull(contours.size());
for (int i = 0; i < contours.size(); i++)
        convexHull(Mat(contours[i]), hull[i]);

//-----Mencari Hull Terbesar
int largest_hull_index = 0;
int largest_hull_area = 0;
for (int i = 0; i < hull.size(); i++)
        if (contourArea(hull[i]) > largest_hull_area)
        {
                largest_hull_area = contourArea(hull[i]);
                largest_hull_index = i;
        }

ball_threshold = Scalar(0);
drawContours(ball_threshold, hull, largest_hull_index,
```

```
Scalar(255), -1);

cvtColor(ball_threshold, ball_threshold, CV_GRAY2BGR);
bitwise_and(frame, ball_threshold, ball);
cvtColor(ball_threshold, ball_threshold, CV_BGR2GRAY);

if (hull.size())
{
        status_bola = 1;

        // ***** Menggambar lingkaran bola pada display
minEnclosingCircle(hull[largest_hull_index], ball_center,
ball_radius);
circle(display, (Point)ball_center, (int)ball_radius,
Scalar(255, 0, 0), 2);

// ***** Posisi bola terhadap vision robot
bola_x_pada_frame = ball_center.x - center_x;
bola_y_pada_frame = center_y - ball_center.y;
sudut_bola = atan2f(bola_y_pada_frame, bola_x_pada_frame)
* 57.295779513;

if (sudut_bola > 180) sudut_bola = sudut_bola - 180;
else if (sudut_bola < 0) sudut_bola = sudut_bola + 180;

// ***** Posisi bola terhadap lapangan
float jarak_bola_ke_robot_frame =
sqrtf(powf(bola_x_pada_frame, 2) + powf(bola_y_pada_frame,
2));
float jarak_bola_ke_robot_lapangan =
pixel_to_cm(jarak_bola_ke_robot_frame);

bola_x_pada_lapangan = posisi_x +
jarak_bola_ke_robot_lapangan * sinf(ofDegToRad(sudut_bola
+ gyro_derajat));
bola_y_pada_lapangan = posisi_y +
jarak_bola_ke_robot_lapangan * -cosf(ofDegToRad(sudut_bola
+ gyro_derajat));

// ***** Cari kecepatan bola, TA KAMAL
timenow = ofGetSystemTime();
bola_x1 = bola_x_pada_lapangan;
bola_y1 = bola_y_pada_lapangan;

if (timenow - timeprev > 150)
{
kecepatan_bola_x = bola_x0 - bola_x1;
kecepatan_bola_y = bola_y0 - bola_y1;
```

```
        float m = kecepatan_bola_y / kecepatan_bola_x;
                if (kecepatan_bola_y < -2)
                {
                kecepatan_bola = -1 * sqrtf(powf(kecepatan_bola_x,
                2) + powf(kecepatan_bola_y, 2));
                }
                else if (kecepatan_bola_y > 2)
                {
                kecepatan_bola = sqrtf(powf(kecepatan_bola_x, 2) +
                powf(kecepatan_bola_y, 2));
                }
                else kecepatan_bola = 0;
                timeprev = timenow;
                bola_x0 = bola_x1;
                bola_y0 = bola_y1;
                }

        }
        else
        {
                status_bola = 0;

                sudut_bola = 90;
                bola_x_pada_frame = 0;
                bola_y_pada_frame = 0;
                bola_x_pada_lapangan = posisi_x;
                bola_y_pada_lapangan = posisi_y;
        }
}

void vision::select_obstacle()
{

        // ***** Membuat threshold obstacle dari threshold
        lapangan dan bola
        bitwise_or(raw_field_threshold, raw_ball_threshold,
        obstacle_threshold);
        bitwise_not(obstacle_threshold, obstacle_threshold);
        bitwise_and(field_threshold, obstacle_threshold,
        obstacle_threshold);

        erode(obstacle_threshold, obstacle_threshold,
        getStructuringElement(MORPH_ELLIPSE, Size(11, 11)));
        circle(obstacle_threshold, (Point)ball_center,
        (int)ball_radius, Scalar(0), -1);

        //-----Mencari Contour
        vector<Mat> contours;
```

```
        vector<Vec4i> hierarchy;
        findContours(obstacle_threshold, contours, hierarchy,
        RETR_TREE, CV_CHAIN_APPROX_SIMPLE);


        obstacle_threshold = Scalar(0);
        for (int i = 0; i < contours.size(); i++)
                if (contourArea(contours[i]) > 1280)
                drawContours(obstacle_threshold, contours, i,
                Scalar(255, 255, 255), 2);

        for (int i = 0; i < contours.size(); i++)
                if (contourArea(contours[i]) > 1280)
                drawContours(display, contours, i, Scalar(0, 0,
                255), 2);

}


//================================================================
=============

void vision::show_display()
{
        line(display, Point(center_x, center_y), Point(center_x -
cosf(gyro_radian) * 60, center_y - sinf(gyro_radian) * 60),
Scalar(255, 255, 255), 5);

        static int fps_20[20];
        float sum = 0;
        float avg = 0;

        fps_1 = ofGetSystemTime();
        fps = 1000 / (fps_1 - fps_0);
        fps_0 = fps_1;

        //Menggeser nilai fps dan menambahkan fps baru
        for (int i = 0; i < 19; i++)
                fps_20[i] = fps_20[i + 1];
        fps_20[19] = fps;

        //Menghitung moving average dari fps
        for (int i = 0; i < 20; i++)
                sum += fps_20[i];
        avg = sum / 20;

        char buffer[33]; sprintf(buffer, "%02.1f FPS", avg);
```

```
        putText(display,       buffer,       Point(10,       600),
FONT_HERSHEY_DUPLEX, 1, Scalar(0, 0, 255), 2);
        sprintf(buffer, "v = %02.1f", kecepatan_bola);
        putText(display,       buffer,       Point(10,       630),
FONT_HERSHEY_DUPLEX, 1, Scalar(0, 0, 255), 2);
        imshow("Display", display);

}

//================================================================
=============

float vision::pixel_to_cm(float _pixel)
{
        return 6.897 * exp(0.0156 * _pixel);
}

float vision::pixel_to_cm_front(float _pixel)
{
        return (-0.0000003 *pow(_pixel,3) - 0.003*pow(_pixel,2) +
2.4561*_pixel - 116.04);
}
```

BIODATA PENULIS

Kamal Arief lahir di Surabaya 1 Januari 1995 merupakan anak kedua dari 2 bersaudara dari pasangan Saleh Assegaf dan Hilmiyah. Penulis menghabiskan banyak kehidupan di Surabaya sejak lahir. Penulis menyelesaikan pendidikan dasar di SDN Kertajaya XIII Surabaya pada tahun 2007 dialnjutkan dengan pendidikan menengah di SMPN 12 Surabaya pada 2010 dan SMAN 2 Surabaya pada tahun 2013. Penulis memulai pendidikan di Institut Teknologi Sepuluh Nopember pada tahun 2014. Selama mengenyam pendidikan disana penulis aktif mengikuti kegiatan robotika ITS dan sempat menjadi tim inti pada ajang KRPAI 2015, KRPAI 2016, KRSBI Beroda 2017, KRSBI Beroda 2018. Selain itu penulis juga aktif menjadi asistem di lab Elektronika Dasar.

Email:
kamalarief17@gmail.com

Halaman ini sengaja dikosongkan